



National Library
of Canada

Bibliothèque nationale
du Canada

Acquisitions and
Bibliographic Services Branch

Direction des acquisitions et
des services bibliographiques

395 Wellington Street
Ottawa, Ontario
K1A 0N4

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Votre lieu - Votre téléphone

Our file - Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

World Information Tool
A geographical approach to resource
discovery on the Internet

by
Luminita Stancu

School of Computer Science
McGill University, Montréal

November 1994

A thesis submitted to the
Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for
the degree of Master of Science

Copyright © 1994 by Luminita Stancu



National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file *Votre référence*

Our file *Notre référence*

THE AUTHOR HAS GRANTED AN
IRREVOCABLE NON-EXCLUSIVE
LICENCE ALLOWING THE NATIONAL
LIBRARY OF CANADA TO
REPRODUCE, LOAN, DISTRIBUTE OR
SELL COPIES OF HIS/HER THESIS BY
ANY MEANS AND IN ANY FORM OR
FORMAT, MAKING THIS THESIS
AVAILABLE TO INTERESTED
PERSONS.

L'AUTEUR A ACCORDE UNE LICENCE
IRREVOCABLE ET NON EXCLUSIVE
PERMETTANT A LA BIBLIOTHEQUE
NATIONALE DU CANADA DE
REPRODUIRE, PRETER, DISTRIBUER
OU VENDRE DES COPIES DE SA
THESE DE QUELQUE MANIERE ET
SOUS QUELQUE FORME QUE CE SOIT
POUR METTRE DES EXEMPLAIRES DE
CETTE THESE A LA DISPOSITION DES
PERSONNE INTERESSEES.

THE AUTHOR RETAINS OWNERSHIP
OF THE COPYRIGHT IN HIS/HER
THESIS. NEITHER THE THESIS NOR
SUBSTANTIAL EXTRACTS FROM IT
MAY BE PRINTED OR OTHERWISE
REPRODUCED WITHOUT HIS/HER
PERMISSION.

L'AUTEUR CONSERVE LA PROPRIETE
DU DROIT D'AUTEUR QUI PROTEGE
SA THESE. NI LA THESE NI DES
EXTRAITS SUBSTANTIELS DE CELLE-
CI NE DOIVENT ETRE IMPRIMES OU
AUTREMENT REPRODUITS SANS SON
AUTORISATION.

ISBN 0-612-05634-1

Canada

*Knowledge is of two kinds. We know the subject ourselves,
or we know where we can find information upon it.*

SAMUEL JOHNSON (1709-1784)

Abstract

The Internet has been conceived as a communication tool to cross geographical boundaries and allow knowledge to be shared among people from all over the world. As a consequence of the expansion of computer networks, the abundance of available information and its variety make the process of *resource discovery* crucial for many people coming from different backgrounds, who often do not have computer skills.

Resource Discovery Services (RDSs) have been developed along with almost every kind of resource, in order to enable access to each specific information pool. Among these services, the most popular are Archie, WAIS, Gopher and WWW, each performing better certain functions rather than others, even though their functionality sometimes overlaps. Inexperienced users are seldom aware of the efficiency of a particular service given a certain task and even less willing to use different tools, each one with its own characteristics.

The diversity of the access methods used by RDSs to retrieve various kinds of data files across the network imposed recently a new trend in the design of RDSs, that is their *interoperability*.

This work is concerned with the presentation of the most common RDSs, followed by the design and implementation of a system called World Information Tool (WIT), able to group existing RDSs and offer to the user the capability of exploiting them in a single, composite tool.

The approach considered by WIT involves only the client side of the RDSs model, without requiring any modifications of the existing servers and protocols. Moreover, the geographical display of the servers permits a better use of the network resources, by attempting to spread the workload among different servers and make users choose nearby servers.

The idea behind the graphical interface of WIT has been influenced by the ease of use offered by hypertext, such that users select servers by clicking on their corresponding buttons displayed on the world map, instead of typing host addresses and specific commands.

WIT represents a valuable tool for the Internet resource discovery because it bridges the use of the existent RDSs and provides access to many resources available through the Internet, allowing users to search and access different types of information from a single interface.

Résumé

L'Internet a été conçu comme un outil de communication pour franchir les barrières géographiques et permettre le partage du savoir par des personnes du monde entier. En raison de l'expansion des réseaux informatiques, l'abondance de l'information disponible et sa diversité rend le processus de recherche de ressources crucial pour beaucoup de personnes venant de différents milieux, qui n'ont bien souvent pas de compétences dans le domaine des ordinateurs.

Les Resource Discovery Services (RDSs) ont été développés avec presque tous les types de ressources, afin de permettre l'accès à chaque source spécifique d'information. Parmi ces services, les plus répandus sont Archie, WAIS, Gopher et WWW, chacun remplissant certaines fonctions mieux que d'autres, même si leur fonctionnalités se chevauchent parfois. Les utilisateurs inexpérimentés sont rarement au fait de l'efficacité de tel ou tel service pour remplir une certaine tâche et encore moins motivés pour utiliser des outils différents, chacun ayant ses particularités. La diversité des méthodes d'accès utilisées par les RDSs pour récupérer divers types de fichiers de données à travers le réseau a imposé récemment une nouvelle tendance dans la conception des RDSs: leur interopérabilité.

L'objet de ce travail est la présentation des RDSs les plus répandus, suivie par la conception et réalisation d'un système nommé World Information Tool (WIT) qui est capable de relier les RDSs existants et offre la possibilité aux utilisateurs de les exploiter au moyen d'un outil composite et unique. L'approche retenue par WIT n'implique que le coté client du modèle RDS, et ne nécessite aucune modification des serveurs et protocoles existants. De plus, la présentation géographique des serveurs permet un meilleur usage des ressources réseau en tentant de répartir la charge de travail sur différents serveurs et pousse l'utilisateur à choisir des serveurs proches.

L'idée sous-jacente à l'interface graphique de WIT a été influencée par la convivialité offerte par l'hypertexte, de façon à ce que les utilisateurs sélectionnent les serveurs en cliquant sur le bouton leur correspondant, plutôt que de taper des adresses hôte et des commandes spécifiques au serveur.

WIT représente un outil précieux pour la recherche de ressources sur Internet car il uniformise l'usage des RDSs existants et donne accès à beaucoup des ressources en informations disponibles à travers l'Internet, permettant aux utilisateurs de chercher et d'accéder à différents types d'information depuis une interface unique.

Acknowledgements

This thesis represents the materialization of a long journey, which started on another continent and aimed to accomplish more than a dream.

I have to confess that when I began this work, my attitude was lacking the impartiality and the focus of a good researcher, but I saw then, as I do now, the development of resource discovery on the Internet as a promise to mankind for shared knowledge, inspiration and collaboration. This research involved a challenge that even if it defeated me, it would be less regretful than if I had not dared to take it.

My deepest gratitude goes therefore to my supervisor, Professor Tim Merrett, who had set me free from the bounds of constrained thoughts, encouraging me to think freely and explore new ideas. He has motivated me to overpass my timid behavior by launching new targets for me and tackled with endless patience the goal of making a researcher out of me. For all this and also for his financial support without which I could not complete my degree, I am forever indebted to him.

Many thanks to my dearest lab mates, Heping Shang and Xiaoyan Zhao, who constantly cheered me up and maintained a warm and enthusiastic atmosphere in the aldat lab, where we spent working together much more time than home.

Almost this entire work has been documented with the precious help of the articles made publicly available via Internet by Tim Berners-Lee, Peter Deutsch, Alan Entage, Chris Weider, Brewster Kahle, and many other people who opened new horizons for me with their thorough perspective and advanced vision.

To Luc Boulianne and the system staff I owe prompt replies and continuous assistance, while I was learning things obvious for them, but not for me.

I cannot forget to mention my special friend, Dr. Marco Ramoni, who watched over my every step, always ready to give me a good advice or enlighten me with an invaluable suggestion during long hours of challenging discussions.

I dedicate the result of these years of school to my mother, together with my new acquired life in Canada.

Contents

1	Introduction	1
1.1	Background	1
1.2	Motivation	2
1.3	Outline	4
2	Internet	6
2.1	What is the Internet?	6
2.2	How did the Internet happen?	6
2.3	Internet growth and value	7
2.4	Internet services	8
2.5	Internet future	10
3	Resource Discovery Services	11
3.1	Overview	11
3.2	Taxonomy of Resource Discovery Services	12
3.3	RDSs architecture	13
3.4	Uniform Resource Identification	15
3.4.1	Uniform Resource Identifiers	15
3.4.2	Uniform Resource Locators	15
3.4.3	Gateways	18
3.4.4	Uniform Resource Names	19
3.4.5	Resource Locators Producers	22

4	Archie	23
4.1	Overview	23
4.2	User's View	25
4.3	Information Provider's View	29
5	WAIS	31
5.1	Overview	31
5.2	User's View	33
5.3	Information Provider's View	35
6	Gopher	38
6.1	Overview	38
6.2	User's View	42
6.3	Information Provider's View	46
7	WWW	48
7.1	Overview	48
7.2	User's View	52
7.3	Information provider's view	56
8	World Information Tool	58
8.1	Overview	58
8.2	Functionality	59
8.3	Architecture	60
8.4	Interface	65
8.5	Design Issues	67
8.6	Implementation	71
8.7	Maintenance	74
9	User's Manual	77
9.1	Getting started	77
9.2	Search strategies	79
9.2.1	Search on keyword	80
9.2.2	Search on service	83
9.2.3	Search on location	86
9.3	RDS Interactions	87

10 Conclusions	92
10.1 Contributions	92
10.2 Further work	93
A URL syntax in BNF	95
B URN syntax in BNF	98
C Setting up a WAIS server	99
D Setting up a Gopher server	101
E Setting up a WWW server	105
F HTML elements	107
G WAISStation User Manual	110

CHAPTER 1

Introduction

1.1 Background

Internet enables people all around the world to communicate and share information rapidly through file transfer, news, email lists, etc. Thousands of different information pools can be accessed electronically on almost everything, from free software applications to weather forecast, much faster and more efficient than paper distribution. But since the Internet is a network of networks, there is no *one place to go for information on what is available and how to access it*.

In the abundance of information on the Internet, resource discovery is crucial. The diversity of data types which are made available on the Internet makes difficult to envisage an overall method for accessing and retrieving information objects. Along with almost every kind of resource, an appropriate service has been developed in order to enable access to each specific information pool:

WAIS [1] appeared in conjunction with the development of on-line libraries, acting as a librarian and indexing every word in collections of text files stored on a single machine, so that documents can be found and searched thoroughly over the network.

Gopher [2] was initially developed for the sole purpose of offering campus-wide access to information residing on different machines and presenting users with menus from which they could navigate across the network. Later on, the Gopher servers spread over the Internet at more than one thousand sites and as a

consequence of this expansion, Veronica has been envisaged as an additional service to help users search through the Gopher menu titles.

WWW [3] was designed as a collaborative tool for the CERN community, based on the hypertext paradigm which links text documents to other documents residing on remote machines. WWW has also introduced the utility of having Universal Resource Identifiers independently of the underlying access protocol.

Learning the Internet requires some commitment and the diversity of information on the network is relevant to a large number of people who are inexperienced with computers. Until the last few years, the Internet was the sole territory of researchers and computer scientists who had neither the interest, the need, nor the time to make friendly user interfaces. The concern for user-friendliness in the process of resource discovery on the Internet dominated the development and the design of WIT, starting from the certitude that one does not need to be a computer expert to use the applications or understand the concepts.

1.2 Motivation

Despite the explosive growth of the network, resources on the Internet are not formally advertised: many hosts provide open access to a variety of documents, but administrators on these hosts usually do not register this information in a directory. And even if one knows what he wants, how does he get it?

Inexperienced users often feel puzzled about where to look for a document or are frustrated when trying to get a reference about a specific topic and find dozens of articles which have no relation with what they are looking for.

Locating a resource is very often difficult due to the cognitive overhead deriving from two common reasons:

- Often users do not know which service is most appropriate for the task. Although some tools provide similar functionality, one of them might be more efficient for a certain task rather than the others.
- Sometimes, a service may burden users with extraneous information. For example, in response to a search for a particular file, Archie provides a list of copies from all over the world. Often the list is too long for the user to scan it. Even

if the list contains only few entries, the user should choose a nearby site from which to retrieve the file in order to make best use of the network.

WIT was born from both the necessity of having a system capable of using various RDSs to search and retrieve information and to increase the awareness of novice users by offering an easy-to-manipulate tool.

Central to this work is the concept of *resource discovery* which implies the act of locating and accessing information resources on the Internet. WIT offers an alternative to the most popular Resource Discovery Services (RDSs), bridging the gap between them and assisting naive users in finding a path toward the information available and manipulating the tools needed to get it.

The diversity of the access methods used by each service to retrieve various kinds of data files across the network imposed recently a new trend in the design of RDSs, that is their *interoperability*. This issue is not easy to accomplish as the variety of existent protocols requires specific parameters and the already built servers have a different command syntax. Therefore, in order to avoid the modification of the existent protocols and servers, the interaction of these services has to be regarded at a high level of their architecture, as they are all based on the client-server model. Thus, WIT considers a graphical approach to this problem, at the client level, making use of the underlying access methods supplied by each incorporated service.

Moreover, the representation of the Internet addresses as dynamic buttons on a geographical map in response to a query is worthwhile not only because it provides a user-friendly interface, but also because it offers more control to the user over the network, based on the distributed servers concept, and in this way she may be able to:

- decrease the total network traffic by attempting to connect to a nearby server;
- balance the workload among the different servers by having alternative, equivalent sources of information.

While most RDSs do not take into account the network traffic and bandwidth, WIT offers the possibility of better using the Internet by choosing more appropriate servers corresponding to the user's needs.

Although the huge carrying capacity of the optical fiber used for communication links gives hope for abundant and cheap data transfers in the future, for the moment

being bandwidth still represents a network resource to be taken into consideration in order to avoid congested links [4].

When using an Internet service, the most natural approach toward saving network resources and achieving a faster response time is minimizing the distance which separates the local machine from the remote server [5], similar to the long distance phone calls which are more costly as their destination is further.

Moreover, when a certain server is not available or its use has a limited users number (for example, try connecting via anonymous FTP to `csd4.csd.uwm.edu`, which has a maximum limit of 20 users), need is to find an alternative server containing the same information.

When looking for copies of a file stored on several machines all over the world, a user from Canada is unlikely to connect to Japan to retrieve that document if another copy is available on a host located in US also because, although the file can be stored under the same name, the copy residing in Japan might be available in Japanese!

Therefore, the user must be able to make a decision as about where to connect when a default server (usually chosen by the system administrator to be the closest one), happens to be unavailable. The Internet address of a machine may provide sometimes information about the location of that host, when its domain part is composed from the code of the country (eg. `clouso.crim.ca` is situated in Canada) or it contains a hint about the site (eg. `evoserv.univ-lyon1.fr` represents the address of a machine at the University of Lyon, France), but most of the time it gives a very broad approximation of the geographical area where the machine is situated and only few people are able to “decipher” it.

As the network size increases from Local Area Networks (LAN) to Wide Area Networks (WAN), the problem of resource discovery changes because of the different techniques in advertizing, offering, locating and accessing the information[6]. Polling or sequential search techniques, in which a user has to search through all potential information sources, become infeasible when there are hundreds, or even thousands of resources.

1.3 Outline

This thesis is structured as it follows:

Chapter 1 is the present chapter, which sketches the background and presents the main reasons leading to the development of WIT.

Chapter 2 introduces the Internet, a “network of networks”, its present growth and future development.

Chapter 3 examines the RDSs, together with the access methods currently used by them to locate and access the information.

Chapter 4 describes Archie, an archive index for thousands of anonymous FTP archives spread world-wide.

Chapter 5 describes WAIS, an indexing and full-text search service for large collections of documents stored locally.

Chapter 6 describes Gopher, a distributed information browser which gives access to various types of information items.

Chapter 7 describes WWW, a hypertext system for browsing distributed information.

Chapter 8 presents the World Information Tool (WIT), a system meant to automate the process of information discovery on the Internet by bringing together the most popular RDSs: Archie, WAIS, Gopher and WWW.

Chapter 9 comprises the User’s Manual of WIT, illustrating each function with examples.

Chapter 10 examines the conclusions that may be derived from this research, the contributions brought by WIT and some suggestions for future work.

CHAPTER 2

Internet

2.1 What is the Internet?

The Internet is a global network of networks enabling computers of all kinds to directly and transparently communicate and share services throughout much of the world. Because the Internet is an enormously valuable, enabling capability for so many people and organizations, it also constitutes a shared global resource of information, knowledge and means of collaboration among countless diverse communities.

There are basically two kinds of access to the Internet:

- *host access* where end-users connect their computers to become part of the Internet, or
- *terminal access* where end users connect to a host computer which is directly connected to the Internet.

2.2 How did the Internet happen?

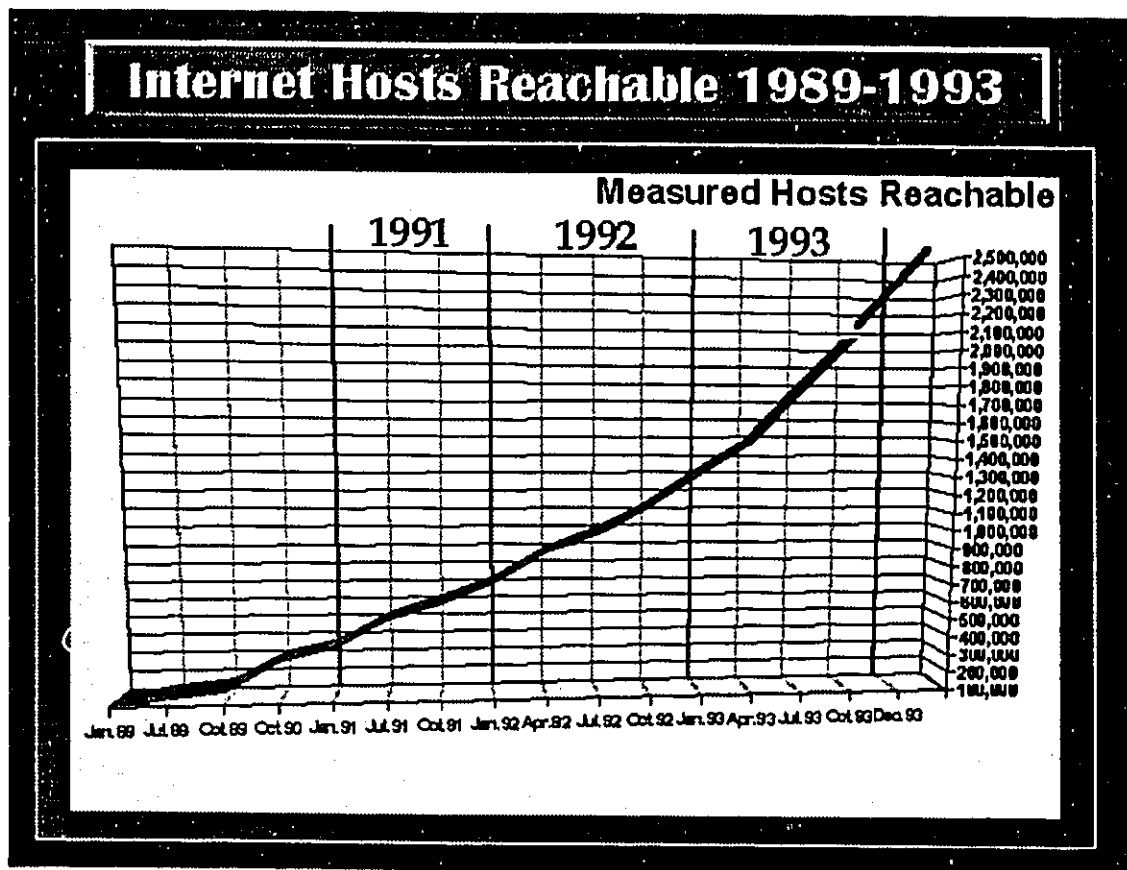
The Internet technology and networks were originally developed by the Advanced Research Projects Agency (ARPA) of the USA Department of Defense (DoD) in the early 1970s to provide robust interconnection of its information resources and researchers.

During the 1980s, the technology and networks were adopted by other government agencies and countries, as well as the private business and the educational-research

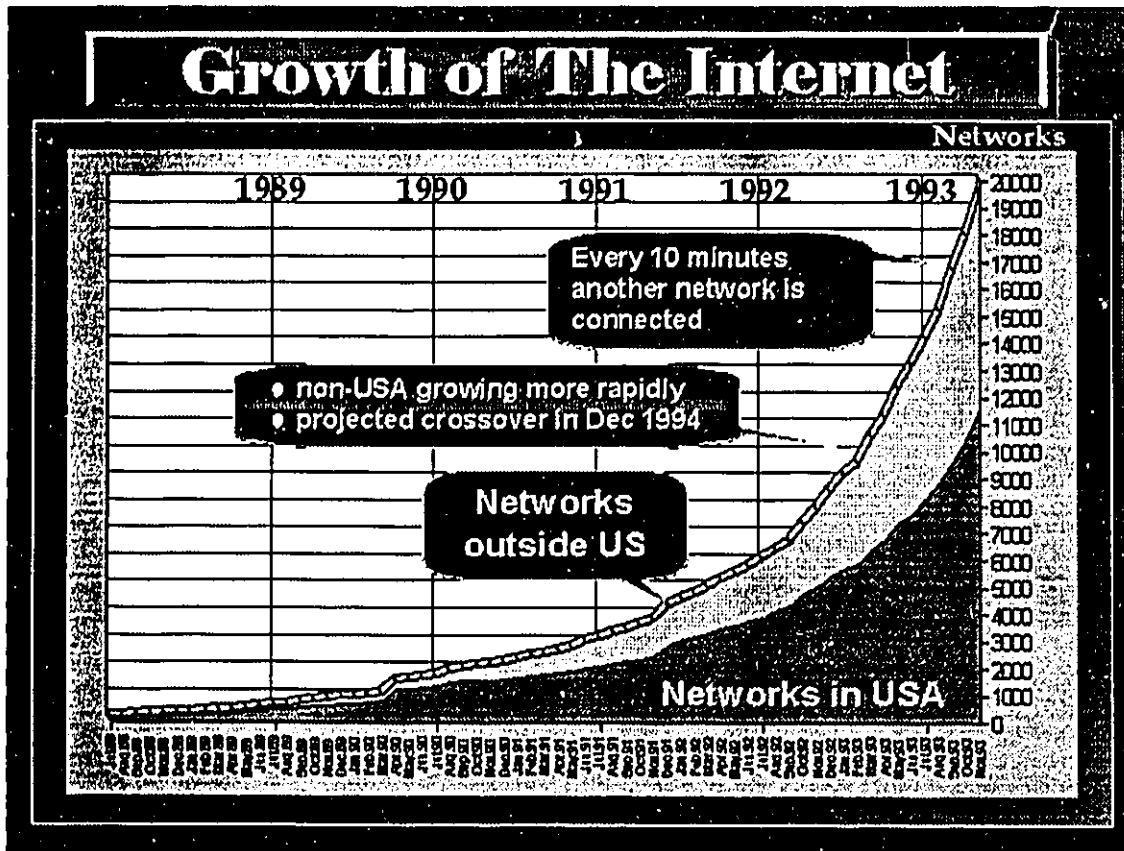
sector. Today, the Internet technology and the Internet have found massive acceptance and use by tens of thousands of organizations around the world.

2.3 Internet growth and value

In April 1994, the Internet consisted of more than 30,000 networks in 71 countries. At the end of 1993, over 2 million computers were measured as actually reachable with an estimated total of 20 million users [7].



The Network growth continues at around 10 percent per month[8], as illustrated in the following picture:



2.4 Internet services

Internet services number in the hundreds, and depend upon a combination of the access computer software and the available bandwidth. The most common services are file transfer, electronic mail (email) and remote computer access (see Figure 2.1 [9]).

Among these services, the File Transfer Protocol (FTP) is by far the most used because it lets users retrieve and put files on remote hosts on which they have an account and it requires little setup; in fact, most vendors deliver machines with FTP configured.

Internet includes also some public services to spread, locate and access information on the network, called Resource Discovery Services (RDSs). The most popular RDSs

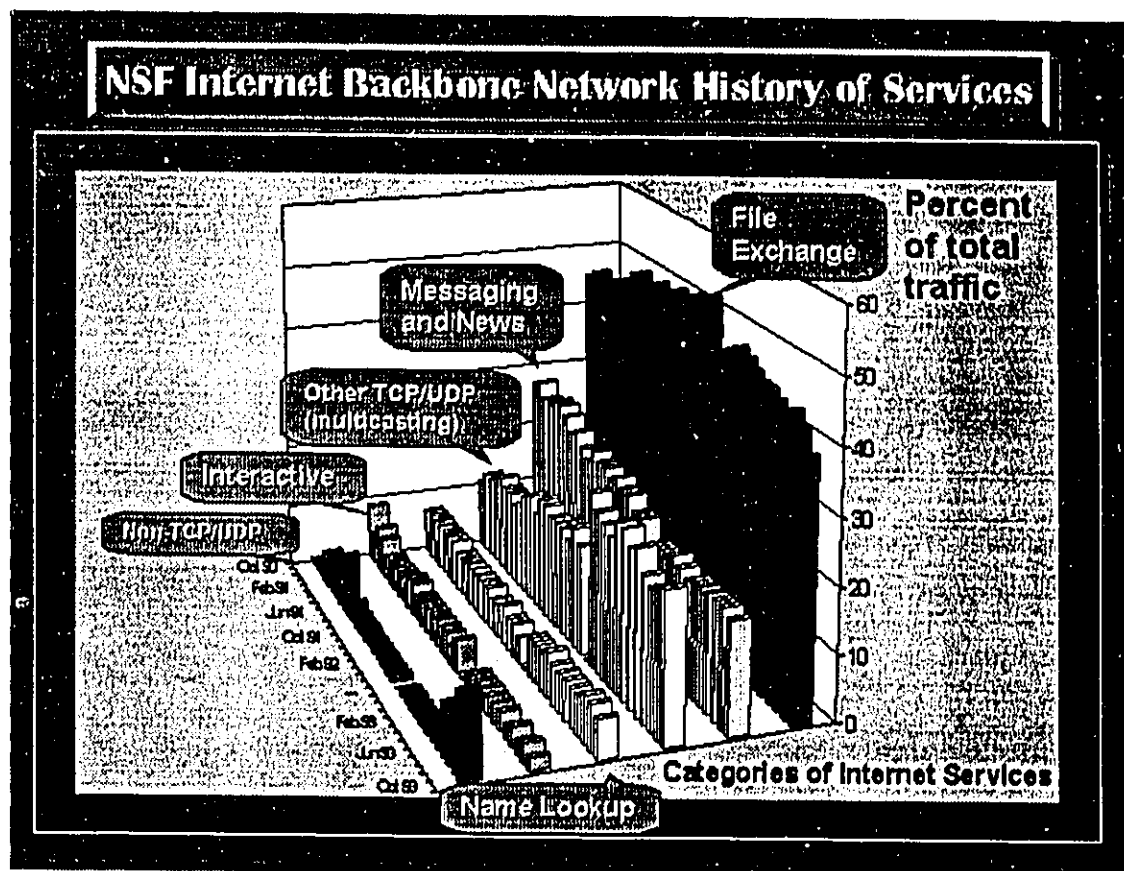


Figure 2.1: Internet Services

are Archie¹, Wide Area Information Server (WAIS), Gopher² and World Wide Web (WWW³).

Other public Internet services include automatic information delivery via Distribution Lists and Netnews; real-time written interactions using Talk or Internet Relay Chat; Netfind, a directory lookup service to discover the network addresses of people and computers and much more.

A taxonomy of the Internet services could be the following:

- **Communication Services** such as: email, mailing lists and news;
- **Resource Sharing Services** such as: file transfer and remote computer access;

¹Name derived from "archives"

²Named derived from "go-pher", someone who fetches necessary items from many locations.

³Sometimes called W3

- **Resource Discovery Services for :**

- people;**

- documents;**

- network resources;**

Unfortunately, it is not easy to keep track of all the new developments in networking. According to some recent estimates, the amount of traffic on the Internet has been increasing ten percent per month, and the number of new applications and services has been growing almost as quickly.

2.5 Internet future

Perhaps the ultimate value of the Internet, however, is enabling communication among millions of people and organizations who can be reached through the network, or who provide abundant and diverse information and software on Internet hosts. For many educational, research, business, military, and governmental activities today, the Internet is an indispensable tool.

In this environment, resource discovery is an increasingly important issue because if users cannot find information, they cannot benefit from the whole potential of the network. Simple software and inexpensive access to the general public over nearly every telecommunications medium are becoming widely available. Almost every conceivable non-profit and for-profit use is underway or being envisioned.

The Internet can shrink the world and bring knowledge, experience, and information on nearly every subject imaginable. A stake in its success is the possibility of finding out requested information on the Internet sea of documents. The Internet future rests with the collaborative work and sources of information over the networks, including the exchange of information across any country boundaries.

CHAPTER 3

Resource Discovery Services

There is so much information distributed on the network, that it is impossible to know where everything is, or even how to begin looking. The large amount and the complexity of the information available on the network motivated the development of **Resource Discovery Services** (RDSs), in order to make the access and the use of this information easier.

3.1 Overview

Within the past three years there has been a widespread adoption of tools such as Archie, WAIS, Gopher, WWW, Hytelnet, Netfind, and many others. The RDSs described in more detail here include Archie [10], WAIS [1], Gopher [2] and WWW [3] as they are being used on a wide variety of information sources including files and databases. Moreover, they are freely available to the networked Research and Education community.

The number of RDSs is large and growing quickly. Certain techniques reappear regularly and seemingly different tools may perform similar tasks: query, browse, retrieval, etc. But even for those services that are provided by different tools, the differences in architecture make some services more suitable for certain tasks, and less efficient or appropriate for others.

The efficiency of retrieval is, in this case, crucial for the overall performance of the whole network, because the access to common sources of information make the user responsible for the time and the resources used.

Unfortunately, users are not always aware of the features of each single service and they overload some servers with steady use, just because novice users have inertia in using a known service rather than finding another one which could be more efficient given a certain task.

It should be noted that in many ways networked information retrieval is in its infancy compared with traditional information retrieval systems. Thesaurus construction, boolean searching and classification control are issues which are under discussion for the popular RDSs, but as yet are not in widespread use. However, with the vast amount of effort that is currently going into the RDSs field, rapid progress is being made. Much work is being done on expanding some of the RDSs to include handling of multimedia information services.

3.2 Taxonomy of Resource Discovery Services

There are two strategies of locating information on the Internet using RDSs: browsing and searching. According to this dichotomy, RDSs can be roughly classified into two categories:

Indexing services, such as Archie and WAIS, organize searchable information into indexing databases and respond to user queries by searching their databases for relevant information.

Browsing services, such as Gopher and WWW, organize their information space as a directed graph with nodes connected by links. They allow users to find data of interest by navigating the information space.

In general, indexes make a system efficient to search, but they do not provide explicit links between related items. In contrast, browsing services offer no easy means of flat searches.

To some tools there had been added in time other services to make the access to their resources easier and they became intermediate systems according to the above dichotomy, having both browsing and indexing features, such as Gopher, for which Veronica¹ has been developed more recently.

The features of the RDSs presented here are summarized in Table 3.1. This taxonomy is not meant to be exhaustive: in addition to the main features mentioned here,

¹See Chapter 6 about Gopher

there are some other characteristics of the RDSs, the *granularity* for example, which indicates if whole documents are being accessed¹, as in the case of Archie and Gopher, or if portions within documents can be pointed at as well, as in the case of WAIS and WWW. Some RDSs also distinguish between *data*, such as files, and *meta-data*, such as indexes or directories.

	Query	Retrieve	Browse	Information Organization	Information Types
Archie	Archie	anonymous FTP (<i>not included</i>)	no	indexes filenames, descriptions	text
WAIS	waissearch	wais	no	full-text index with waisindex	text,
Gopher	Veronica	gopher	yes	indexes menu titles with Veronica	multimedia
WWW	-	HTTP	yes	links	multimedia

Table 3.1: RDSs Features.

But even if one knows which resources are available, how can they be used and when is an RDS more advantageous than another?

3.3 RDSs architecture

The architecture of a RDS must cope with a widely distributed heterogeneous network of computers running different applications which use different data formats. This requires a *client-server model* [11].

The client-server model divides the service into three parts. Each of these parts performs a distinct function in the implementation, as follows:

1. The **client** process, which uses the service and makes requests;
2. The **server** process, which provides data in a form negotiated with the client;
3. The **protocol**, which the client and the server use to communicate with each other.

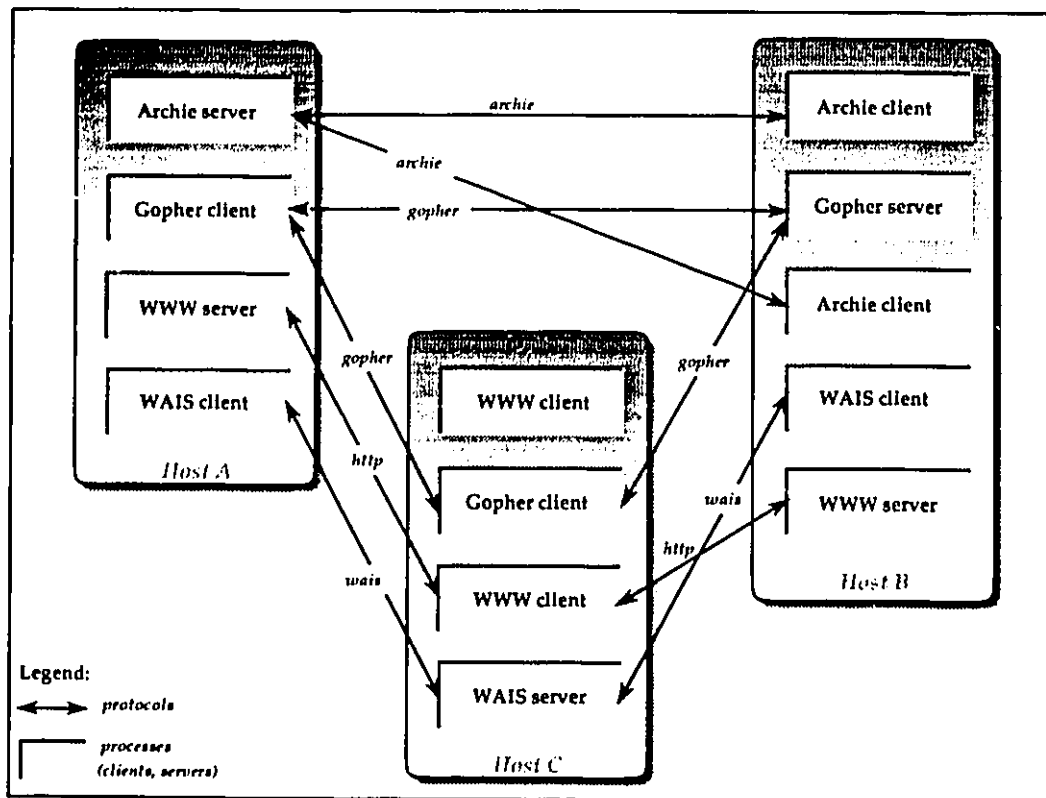


Figure 3.1: The client-server model

A service protocol and a server may be designed to allow many client processes to use the same server process, or a server may serialize client access and fill only one client request at a time.

The client-server model for RDSs described here involves processes running on machines, not on dedicated client or server machines. Each host can have server processes from one or several application protocols and client processes using one or more servers, as shown in Figure 3.1.

A client process can communicate with a server process on any host running the server. Also two hosts can each be running servers for the same application protocol and also each can have client processes connecting to the server of the other one.

3.4 Uniform Resource Identification

As networked information resources proliferate on the Internet, standard means for identifying, locating, and describing these resources become increasingly necessary. Also, many protocols and systems for document search and retrieval are currently in use, and many others are to be expected in a field whose expansion is explosive.

The possibility of locating and accessing the same object by different RDSs relies on the scheme for naming objects. The concept of *object* here implies not only human-readable documents, but a generalized unit of reference and does not need to correspond to any unit of storage. The term *name* represents a string of characters describing an object. A name allows a user to retrieve or operate on objects with the help of a client program via a server program.

The objects on the network which are to be named and addressed include typically objects which can be retrieved and objects which can be searched, but there is a large variety of other objects which may support other operations. To access objects which are part of different information spaces, there is a need for a universal syntax to encapsulate the name of objects. This allows names in different spaces to be treated in a common way, even though names in different spaces have different characteristics, as do the objects they refer to.

3.4.1 Uniform Resource Identifiers

Uniform Resource Identifiers (URIs) aim to enable global search and retrieval of documents across various computing platforms and despite the diversity of protocols and data formats. URIs encode the names and addresses of objects on the Internet.

A URI is an object of a universal set of names in registered name spaces and addresses referring to registered protocols or name spaces [12].

3.4.2 Uniform Resource Locators

A general *resource locator* is an object that describes the location of a resource [13]. An Internet resource locator consists of a service and access parameters meaningful to that service. Each service utilizes a particular addressing scheme.

The concept of Uniform Resource Locator (URL) was introduced by Tim Berners-Lee and it was designed to provide a uniform way of specifying location and retrieval information for networked objects. A URL is a form of URI which expresses an address

which maps onto an access algorithm using network protocols [14]. As defined by Tim Berners-Lee, “a URL represents physical addresses of objects which are retrievable using already existing protocols on the Internet” [15].

The generic syntax of URLs, shown below, provides a framework for new names to be resolved using as yet undefined protocols:

`scheme://[username@]IPAddress[:port number]/path`

A *scheme* is a registered identifier for a protocol. Table 3.2 shows the schemes currently covered by the URL syntax [15].

http	Hypertext Transfer Protocol
ftp	File Transfer Protocol
gopher	The Gopher Protocol
mailto	Electronic mail address
mid	Message identifiers for electronic mail
news	Usenet news
wais	Wide Area Information Servers
telnet	the TELNET Protocol
rlogin	remote login
tn3270	remote login to IBM VM/CMS

Table 3.2: Currently registered schemes used in URLs

The Internet Assigned Numbers Authority (IANA) is the central coordinator that registers and maintains schemes [16]. Any submission of a new URI scheme must include a definition of an algorithm for the retrieval of any object within that scheme. The algorithm must take the URI and produce either a set of URL(s) which will lead to the desired object, or the object itself.

Within the URL of an object, the part of the URL following the name of the scheme is in a format depending on the scheme. To connect to systems which do not use the anonymous FTP convention that specifies the user name is “anonymous” and the password is the user’s Internet mail address, the syntax allows the inclusion of a user name and of a password. The IPaddress represents the Internet address of the server host, and it is followed by the port number, which is used in case the

port is not the default one. The rest of the locator is known as the *path* and it has a significance defined by the particular scheme. It defines details of how the client should communicate with the server, including information to be passed transparently to the server without any processing by the client. Typically it is used to encode a name in a given name space or an algorithm for accessing an object. Here are presented some examples of URLs:

- An example illustrating an FTP scheme that identifies an image file on the Internet host `ftp.tic.com`:

```
ftp://ftp.tic.com/matrix/maps/matrix/world.gif
|         |         |-----|
scheme IPAddress                path
```

The port number is omitted because it is used the default one.

- The following example refers to a WWW scheme identifying a hypertext document called “Overview.html” on the WWW server `info.cern.ch`:

```
http://info.cern.ch:80/hypertext/DataSources/Overview.html
```

- The next example refers to a Gopher scheme that results in the identification of the Gopher item called “Information About Gopher”. The percent sign (%) is used as an escape character in the encoding scheme, while 20 represents the blank spaces between words in the Gopher menu title:

```
gopher://gopher.micro.umn.edu:70/11/Information%20About%20Gopher
```

- The example below refers to a WAIS scheme which identifies the WAIS source called “directory-of-servers” and performs a search for the word “versions” on it. The question mark (?) is used to delimit the boundary between the URL of an index and a set of words used to express a query on that index:

```
wais://cnidr.org:210/directory-of-servers?versions
```

In all cases the client passes the path string uninterpreted to the server. For the URL syntax in BNF format, see Appendix A.

From the client-server model point of view, the RDS architecture must separate the location and access components from the identification of an information object in order to permit an interaction between different systems that access the same data. Therefore, the client has the responsibility of resolving an object address into an object using its repertoire of network protocols, while the server provides the location and the protocol offers access to the information object. A document name provides a way for the client to find the server and for the server to find the document. Even portions of a document may be specified using the URL syntax, by separating the fragment identifier from the whole object with a hash sign (#), as shown in the following example:

```
http://info.cern.ch:80/hypertext/WWW/Addressing/BNF.html#21
|           |           |           |           |
scheme  IPaddress port           document fragment-id
```

There is a trade-off between implementing the fragment retrieval as a part of the service provider or as a function of the client: if only a small part of a large object is needed by the client, it does not need to transfer the entire object across the network. However, the implementation of this function division requires that fragment specification be passed to the locator service provider as an additional parameter. This complicates the service definitions and requires the service to return a much more complex series of error conditions pertaining to invalid or unsupported fragment specifications. In the example considered above, the reference to the particular portion of the document `BNF.html` identified by the identifier `#21`, is not sent to the server, but it is retained by the WWW client and used when the whole object has been retrieved.

3.4.3 Gateways

To access objects in another RDS information space, some clients are configured to use specific servers, called *gateways*, that translate locators for indirect access through other addressing schemes, while other clients may have incorporated the function of converting locators themselves.

Example: The Mosaic client for XWindows can access a WAIS server by entering a URL with the following syntax:

`waiss://<waiss_server>:<port>/<source_name>`

In this URL `<port>` represents the port number the WAIS server is running at, and `<source_name>` is the name of the WAIS database. Consider for instance the following locator:

`waiss://cnidr.org:210/directory-of-servers`

What this means is that given the name of a scheme different than the http protocol used by Mosaic, the client is configured to use specific gateway locators for indirect access through another scheme, and converts a URL like that into a URL that looks like the following:

```
http://www.ncsa.uiuc.edu:8001/cnidr.org:210/directory-of-servers
      |_____| |_____|
      HTTP -> WAIS gateway      remains the same as before
```

`www.ncsa.uiuc.edu:8001` is the address of a public WAIS gateway; everything after the first single slash in this URL is exactly the same as in the original URL. This should give the gateway all the information it needs to access the specified WAIS database and provides the non-native-WAIS client with the equivalent of direct access.

3.4.4 Uniform Resource Names

However, URL does not provide a stable, long-lived reference to a resource as the resources can move out from under the locator. Moreover, a given resource may have multiple URLs if it resides at a number of different locations on the net, or is available under a number of different access methods.

The *Uniform Resource Name*, or URN, has been designed to alleviate this problem. A URN is a name which uniquely identifies a resource, and it is designed to provide persistent naming for networked objects [17]. A URN identifies an object uniquely by its content, so the object may be moved from one location to another, or it may reside on multiple locations and these locations may change in time. An International Standard Book Number (ISBN) or a book call number within a given library collection could be a relevant analogy to a URN. A URN is intended to be used in conjunction with a directory service, which can provide a URN→URL mapping. This URN-URL

architecture allows permanent references to be made to resources without caring about their current locations. A possible, but not yet feasible, scenario for the use of the URN→URL mapping is the following:

- A client program, running on the user's workstation, receives a hypertext document, menu, or search result etc., containing a number of URNs.
- The user selects one of the URNs which depicts a relevant object. Let us suppose she chooses:

`urn:iana/jimmy.mcgill:wit_manual`

The URN needs to contain enough information for the client to locate the resolution service.

- The client locates a URN→URL resolution service. This might be done by extracting the ID Authority from the URN. In the example considered here this would be `iana/jimmy.mcgill`, reversed and “.urn” appended to form “`jimmy.mcgill.iana.urn`”. This address might be then passed to the Domain Name Server which can resolve it into an Internet address.
- The client contacts the URN→URL resolution service, and retrieves a number of URLs for this document:

URL: `http://www.mcgill.ca/pub/docs/wit.html`

URL: `ftp://jimmy.cs.mcgill.ca/grads/luminita/wit_manual.ps`

- The user, or the client, chooses the “best” URL. In some cases the URL will contain enough information for the client to fetch the document without further input from the user. For instance, in the example above, if the client does not support hypertext, then it might automatically select the postscript version.
- The client either retrieves the URL itself or, if the URL is for an access method the client knows about, but it cannot handle itself, then it passes the URL to an appropriate gateway. In the case the client has never heard of the protocol, it should hand the URL to its default gateway.

- The client either displays the object, or if it is in a format the client does not handle, it launches an appropriate viewer.

There are three components to the URN:

1. a *wrapper*,
2. a naming *Authority identifier*,
3. and an *opaque name*.

The general syntax of a URN is given below:

URN : <Authority_identifier> : < opaque_name>

The <Authority_identifier> consists of two sub-components, a *scheme identifier* and an *individual identifier*. The individual identifier is the identifier of an organization which has assigned the opaque_name component to the resource and can resolve the URN to a set of URLs for the resource. It can be structured in a hierarchical way and it may be multi-leveled. The scheme identifier and the individual identifier are separated by a colon. A typical naming Authority_identifier looks like the following:

IANA:42117

The <opaque_name> component of the URN is any string the naming Authority wishes to assign to a given resource. A given opaque_name, once assigned, should never be reused because URNs are expected to be *persistent* names for resources. An example might be the following:

```

URN:IANA: 42117: 1.306.457
  |         |         |
  scheme individual opaque
   id      id      name
  |-----|
  Authority_identifier

```

To summarize, a URN identifies a resource or unit of information, while a URL identifies the location or a container of a resource identified by a URN.

3.4.5 Resource Locators Producers

The provider of a resource may produce a locator for it, leaving the locator in places where it is intended to be discovered, such as an HTML page, a Gopher menu, a message to an e-mail list or an announcement on the Internet News.

Users also are producers of resource locators: for instance, sometimes a user composes a resource locator based on an educated guess and submits it to client software with the intent of establishing access. For example, one can find out about the existence of a new WWW server via an email list or by reading the Internet News. S/he extracts the URL and then enters it by hand using the WWW interface to connect to that remote server and get usually its home page ². Consider a concrete example:

By reading the News, one finds the following announcement in the group `comp.infosystems.announce`:

```
From: jameslow@iti.gov.sg (James Low)
Newsgroups: comp.infosystems.announce
Subject: New Web Server [Information Tech Institute, NCB, Singapore]
Date: 8 Aug 1994 08:18:41 GMT
```

The Information Technology Institute (ITI) is the applied R&D arm of the National Computer Board (NCB) of Singapore.

We are pleased to announce our World Wide Web server, which contains info on our R&D programs, fact sheets, the ITI Innovator newsletter, publication lists, conference announcements, press releases, job openings, etc. The URL is <http://www.iti.gov.sg/>

The ITI Web server is continuously being improved. We hope you will visit our Web server. Do provide your valuable feedback if you wish. Thank you.

If s/he wants to read the document posted there, all it takes to obtain the file is to enter the locator specified in the posting, <http://www.iti.gov.sg/>, using a WWW client to retrieve and display the document.

²See Chapter 7 about WWW

CHAPTER 4

Archie

The Archie [10] system, as its name suggests, is a distributed archive index system that gathers, indexes and serves public information available on the Internet.

4.1 Overview

The Archie service began as a simple project to catalog the contents of hundreds of *anonymous FTP* archives sites.

Anonymous FTP is a special FTP service which permits any user on the Internet to access a remote filesystem in order to transfer files to and from that site. It doesn't require the user to have a personal account on the remote machine and therefore it is widely used for making available or getting public information on the Internet.

For anonymous FTP, one has to use an FTP client program to connect to an anonymous FTP server and login as user "ftp" with the password being the e-mail address of the user.

Archie servers gather location information, name, and other details describing FTP-available on-line files and create two index databases:

1. the **filenames** database, which contains the list of names and locations of files at archive sites that provide anonymous FTP access. A Unix shell script using the **cron** command, which executes commands at specified times and dates, is utilized to connect via anonymous FTP to each monitored site in turn and to fetch the local listing of files by performing the FTP command "**ls -lR**" which recursively lists the content of directories. The **filenames** database is basically

obtained by merging the recursive listing of files from all the anonymous FTP sites harvested monthly by Archie.

An example of a local entry in the filenames database is the following [18]:

```
pub/archives/article:
total 15
-r--r--r--  1 root  15277 May 7 1991 archives.article.text.Z
```

2. the **whatis** database contains the names and descriptions of files usually available through anonymous FTP¹. The entries in this database are simple text strings entered by hand. Currently there is no standard for encoding text descriptions at each archive site, but work started to be done on this and once it will be completed, Archie servers will be able to gather and index this information automatically [19]. An entry in this database looks like the following:

```
<filename> <associated description(text string)>
```

Archie *servers* poll thousands of FTP servers monthly, build the “filenames” and “whatis” databases and make them accessible to clients (see Figure 4.1). Archie *clients* send queries to an Archie server which searches its databases and returns the result.

Offering a very popular service, Archie servers are sometimes overloaded by simultaneous requests. In order to protect the responsiveness of the service, limits on the number of simultaneous requests have been set for each server.

Archie servers are set up to respond to smaller requests much faster than to large requests. Therefore, it is important to know how to make the best use of the service, because larger the request, longer the waiting period for everyone’s replies will be.

There are some relevant limitations of Archie:

- There is no way to search for a file in a particular directory.
- If what you found is a directory that might contain an useful file, the only way you can find it out is to use anonymous FTP to see the content of that directory and eventually to retrieve the file.

¹This might not always be the case.

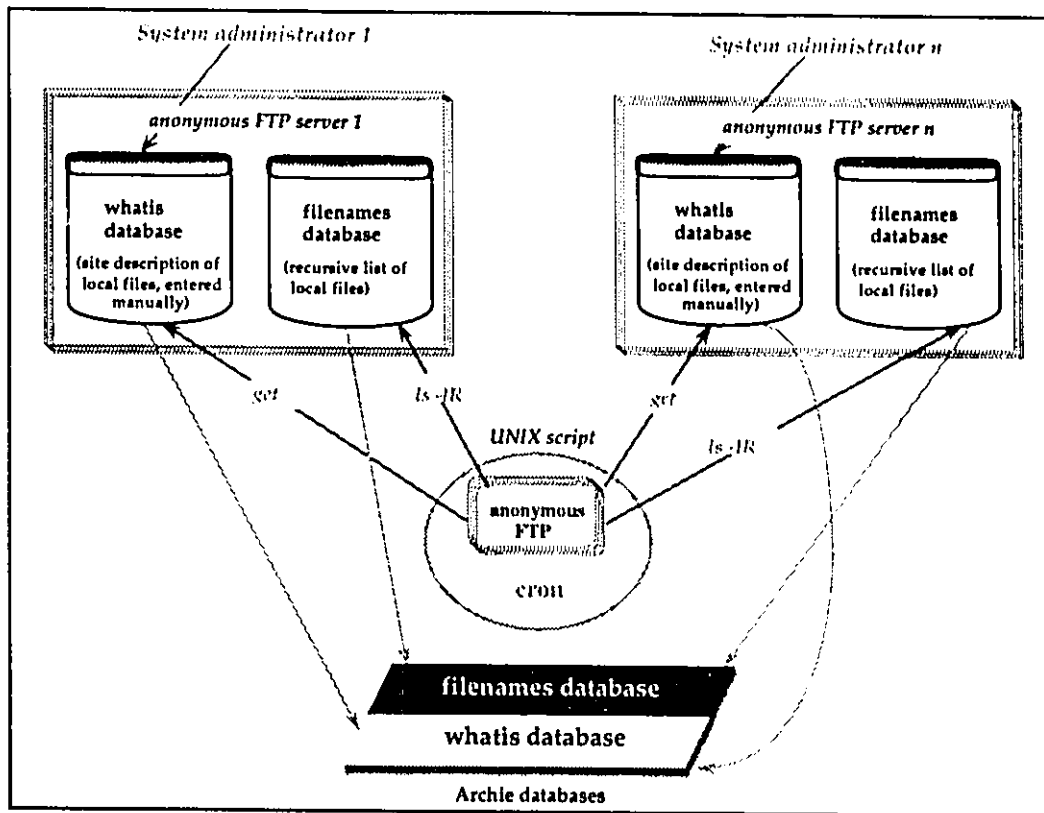


Figure 4.1: The Archie System

- There might be entries in the “whatis” database that are not updated, so in the case of a deletion for example, a file might appear in the Archie reply to a query and not be there anymore.

4.2 User's View

Users can contact an Archie server and search its databases for files they require. Since Archie servers contain the same information, the user's choice of an Archie server is motivated by the geographical location of an available server in order to:

- spread the workload among servers;
- minimize the network traffic.

Archie Server	Suggested area for use
<code>archie.rutgers.edu</code>	Northeastern U.S.
<code>archie.sura.net</code>	Southeastern U.S.
<code>archie.unl.edu</code>	Western U.S.
<code>archie.ans.net</code>	ANS network
<code>archie.au</code>	Australia
<code>archie.funet.fi</code>	Europe
<code>archie.doc.ic.ac.uk</code>	United Kingdom

Table 4.1: Some available Archie servers distributed worldwide

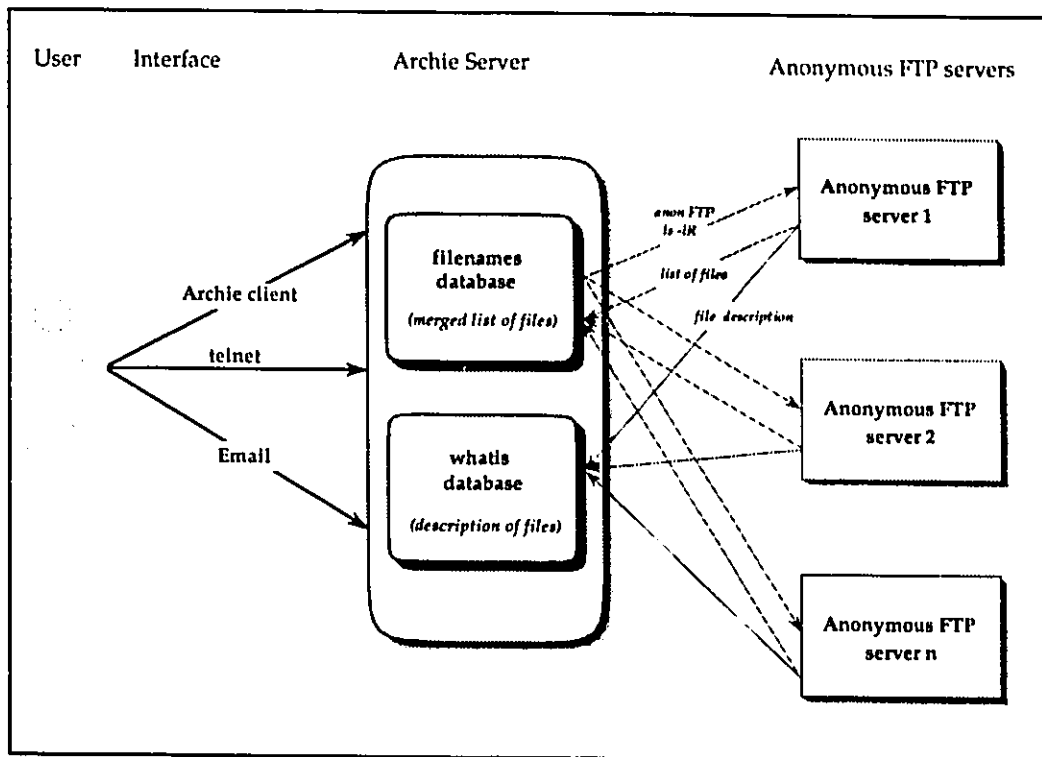
Table 4.1 contains a list of Archie servers and suggested areas for use.

Users can search the Archie databases in three ways:

1. through a **telnet** session to a machine running an Archie server, `telnet archie.sura.net` for example, and log in as user “archie”, without any password. The system usually prints a banner and after that the Archie command prompt. As interactive logins consume network bandwidth and their number is often limited by the server, it is indicated to use a stand-alone client software [20].
2. by using a stand-alone **client program**, which allows users of the Prospero system to access the Archie databases through a Prospero server, without the need to log onto the Archie server directly [19]. Use of client software is encouraged because it makes non-interactive use of the Archie servers and subsequently has a better performance on both sides.
3. through an **email** interface which allows users to send and receive search requests via electronic mail. The e-mail approach uses less network bandwidth than an interactive `telnet` session. The user simply sends a message to an Archie server, `mail archie@<server-address>`, with no subject line and Archie commands in the body of the message [21]. For example, the following represents a request for the list of all sites in Canada who are tracked by the Archie server and then a list of files matching a regular expression of the form `<pattern>` as defined by the standard Unix text editor `ed`:

```
list *.ca
```

```
find <pattern>
quit
```



Users can query the Archie databases for filenames that match:

- specified patterns;
- a list of FTP archive sites;
- a list of files available from specific sites.

Archie does not fetch the files for them, but it indicates the locations and the directory where the files reside. The results returned by the Archie server in response to the search command issued by the user can then be used to fetch the file(s) directly from the archive site using the `ftp` command.

From the user standpoint Archie could be regarded as a "secondary source" of information which, because of the high cost of locating and serving, would not otherwise be available.

Freely available Archie clients exist for most operating systems and can be fetched using anonymous FTP [22]. There are also gateways to the Archie system from many

other RDSs, including Gopher, WAIS and WWW. An X.500 (standard for directory services) interface to Archie is currently under development to provide a White Pages service for locating people on the Internet.

Example Search with Archie for filenames and/or description of files which contain the substring "orbix". To accomplish this task, let's assume the user has an Archie client which connects by default to the closest available Archie server. For substring case insensitive search one has to choose the option `-s`, as follows:

```
> archie -s orbix
```

```
Host clouso.crim.ca
```

```
Location: /igloo/public/outils/information-CORBA
```

```
DIRECTORY drwxr-xr-x      512 Nov 21 00:00 ORBIX
```

```
Host world.std.com
```

```
Location: /pub
```

```
FILE -rw-r--r--      9282 Mar  8 04:27 orbix_from_iona
```

In case a filename or a description of a file matches the search substring, the entry in the Archie reply begins with the word "FILE".

Sometimes the search will match a substring in a directory path, but no filename in that directory. In this case, the location found by Archie shows the path, and the entry listed will start with the word "DIRECTORY". This indicates only that directory might contain something useful, but to actually find it out, the user has to connect there with anonymous FTP and to look into that directory.

Since there might be several anonymous FTP sites that have the same information, a new problem occurs: which one to chose. In this case, the closest anonymous FTP server would be the best choice, but Archie does not help much here, because very few people can figure out a geographical location from the Internet address of a machine.

4.3 Information Provider's View

The Archie system keeps track of anonymous FTP archives on the basis of volunteer labour. The Archie service requires interaction with the anonymous FTP site administrators and information providers have to devote time and space on their machines to make their resources available. There are two types of information providers who would be interested in Archie:

1. Primary information providers are interested in having a summary of the information provided by their service tracked by an Archie server. To simplify the recursive local listing operation, some site administrators prepare this listing in the "ls-lR" file and make it available via anonymous FTP. Also, this kind of information providers have to create and update the "whatis" local database by entering the descriptions of files by hand.

To be tracked by the Archie service, the anonymous FTP site administrator must report the existence of his site to the Archie server site administrator. There is not as yet an automatic mechanism to register new sites [19].

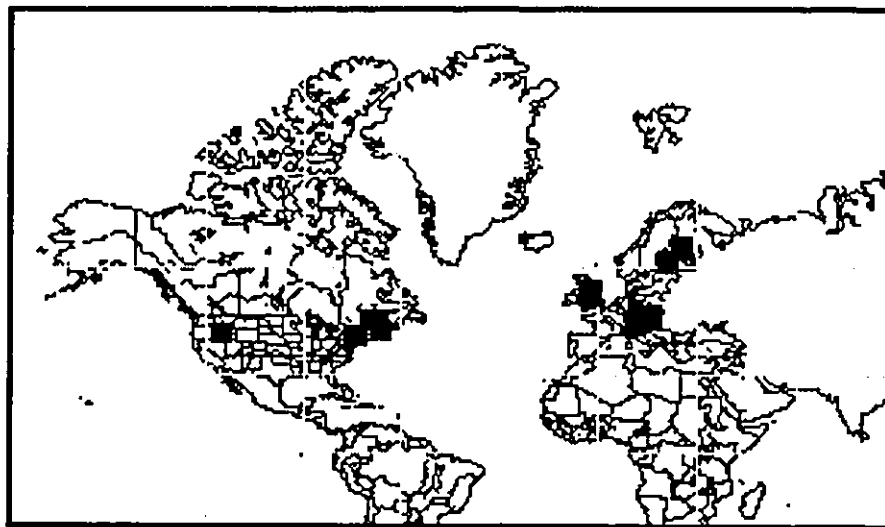
2. Secondary service providers, or those sites wishing to provide a "value-added" service for the Internet, elect to run an Archie server at their site to provide a useful service to users, to raise the profile of their institution on the Internet, or for commercial services.

The prototype Archie server at McGill uses a round-robin algorithm to connect to each monitored FTP site in turn, cycling through the entire list of sites once a month [19]. Other Archie servers limit themselves to mirror the databases gathered by other servers, thus reducing the load on the network connections and sometimes minimizing the user need for further access. There is still work to be done to ensure consistency among Archie servers, that is to get the same responses to the same question.

The Archie system is of particular utility serving information where there are many sites to be searched and/or where the cost of searching each site is high. Searching for a specific filename at a single site may involve scanning hundreds, or even thousands of filenames. Thus, most operators of anonymous FTP archives welcome the fact that Archie indexes and serves the names of all files available from each site tracked.

In the winter of 1992, there were 9 Archie servers and they were tracking approximately 1,500,000 files provided by some 900 anonymous FTP archives, with additional

information being added daily [19]. After just two years of service, in November 1993, Archie was tracking over 2,100,000 filenames on 1,200 registered FTP archive sites around the world [23]. Now the number of Archie servers is 22², and they are spread worldwide, as shown in the picture below:



Work continues on extending the Archie service to provide additional types of information. The latest version is being used to provide a prototype Yellow Pages service and directories of online library catalogs and electronic mailing lists.

²Number based on the list of Archie servers supplied by archie.cs.mcgill.ca when given the command "servers" as of June 27, 1994.

CHAPTER 5

WAIS

The WAIS [1] project was developed by Thinking Machines, Apple Computer, Dow Jones Corporation and KPMG Peat Marwick and initially it was intended for the use of business executives [24]. Subsequently it became an important service which made network-available full-text indexed collections of documents from specific Internet sites.

5.1 Overview

WAIS is an RDS used to index large collections of files resident on a single machine. It is distinct from Archie in that it locates the index and the source files on the same machine, while Archie, as shown, indexes filenames and descriptions of files, polling thousands of servers (see Table 5.1).

The decentralization of the WAIS set of indexes among independent servers that provide the information has two main consequences:

- WAIS has better scalability properties than Archie's global index, meaning that it is easy to add more WAIS servers on the network, as they are independent, but on the other hand,
- it does not provide global search across servers. Instead, users have explicitly to select one or more particular servers to search.

WAIS	Archie
full-text index	filenames index
indexes documents on only one server	indexes thousands of servers
retrieves/displays documents	does not fetch documents
no harvest	monthly harvest
no data duplication	data duplication
no data structures	data structured in fields

Table 5.1: WAIS vs Archie

WAIS uses a single computer-to-computer protocol which is an extended version of the ANSI Z39.50 query and retrieval protocol [25], a NISO standard intended particularly for querying library catalogues.

A WAIS *server* is associated with a specific collection of files called *source* and it allows the set up of a *full-text* index of local documents to be published. The *client* is the user interface and it is capable of translating user requests into the standard protocol used in the system. WAIS servers current information is mostly text based, but this is not a restriction of the protocol since the documents retrieved can be an arbitrary byte stream. For example, the weather server at quake.think.com provides satellite map pictures in GIF format, which can be displayed by WAIS clients [26].

WAIS questions are formulated by the user in natural English language, then the client application translates the queries into the WAIS protocol and transmits them one by one over the network to the server. When the server receives a transmission, it translates it into its own query language, searches the source for documents satisfying the query and marks them for retrieval. The list of relevant documents is then encoded in the response and transmitted back to the client. The client decodes the response and displays the potential results. Matching documents are ranked according to a simple statistical word-weighting scheme which attempts to indicate likely relevance. The user may choose to view selected documents and thus retrieve them across the network, or to find similar documents using the results obtained to rerun the search: this method is called *relevance feedback* [27] and it represents one of the main advantages of WAIS. The system examines the document marked as being relevant, breaks it into words and uses the most important words in the document to

construct a new query. This query will be run against the database and the result will be presented to the user. This process is hidden from the user, who only knows that s/he has indicated to the system that a particular document was interesting and that the system has presented her/him with more documents on the same topic.

5.2 User's View

The easiest way to try out how WAIS works is to connect to a WAIS server using `telnet`, for example,

```
> telnet wais.wais.com
```

and enter at the "login:" the username "wais". This permits the use of a curses-based interface¹, a simple terminal window client program, called *swais* (Simple WAIS interface) [29]. This is a basic access tool designed for those users focused on data retrieval and not computer operation. The functionality supported includes source selection, keyword entry, and automatic document retrieval, but it does not provide relevance feedback or a mechanism for storing questions. In order to take thorough advantage of the features of the system and of a direct-manipulation interface, the user has to employ a more sophisticated WAIS client software, such as the one for NeXT.

The client connects to a particular WAIS server selected by the user's choice of the source, and it specifies a search pattern which is matched against the server's index. Searches are specified as simple natural-language queries; common words such as "the" are removed, and Boolean "ORs" are implicitly added between the remaining list of words.

If the user selects one of the document titles listed in reply, the document is actually retrieved across the network, brought to the local workstation and displayed.

Once the user has found relevant documents, s/he can ask the source(s) for other documents *similar* to the ones found, in order to refine the search. In the present WAIS system, similar documents are simply the ones that share a large number of common words.

The user can start a new query anytime, without abandoning the old one and he can also save a query and its sources. This might be useful, for instance, if the user

¹Curses are screen functions with "optimal" cursor motion [28].

wants to update a response as a database changes: he runs the same query again later on, and this may be the case with searches performed on news archives, for example.

Because every word in a document is indexed, the WAIS search is highly time consuming, so the sources to be queried and the search words must be chosen carefully. On the other hand, the user has no way of telling where a source is located, because the source name is related only to its content.

Both free and commercial WAIS clients are available for various platforms [30]. To use commercial information services, that make their products available through WAIS interfaces, users have to pay a fee. For example, the pay-for server of the Dow Jones Information Service contains several months of the Wall Street Journal and 450 business publications [31].

Example: Suppose the task of looking for documents related to “versions” is given. When using a WAIS client for NeXT, three panels are displayed when the application is launched, as shown in Figure 5.1:

1. the Sources palette;
2. the WAIS Question panel;
3. the Documents Palette;

The user is invited to type a natural language question in the “Tell me about” field of the WAIS Question panel. In this case, s/he obviously types the word “versions”.

Once the query is entered, one or multiple WAIS sources must be selected by dragging the desired source from the Source palette into the appropriate slot, next to the field entitled “Using these Sources”. The user presses then the “SEARCH” button on the WAIS Question panel to perform the search.

The selection of a source assumes that the user has already got some idea where to look for the information needed, or s/he might initially use the special database called *directory-of-servers*, that advertises all publicly available WAIS sources. To do this, the user drags the *directory-of-servers* from the Sources Palette into the sources well, then presses the “SEARCH” button. A list of sources that contain the word “versions” in their description is listed in the field called “Resulting Documents” on the WAIS Question Panel. One can get information about a source by double-clicking on the selected source name and a new panel will be displayed, containing the

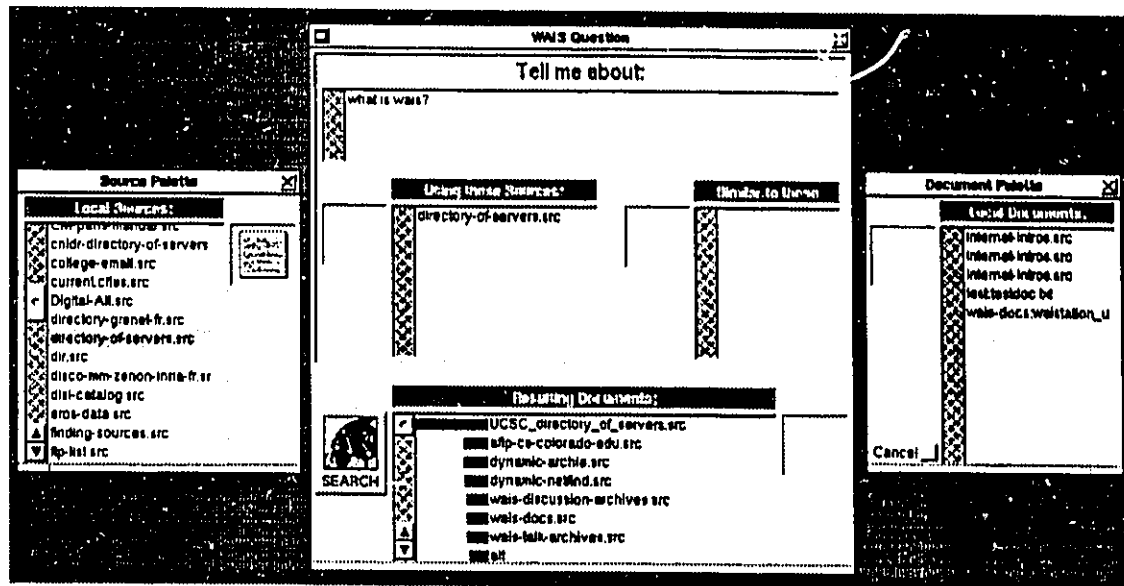


Figure 5.1: The NeXT interface of the WAIS client

description of that source. Selected names, for both sources or documents, become highlighted.

Suppose the user chooses the source called `stsci-docs.src`, then clicks on the “SEARCH” button. WAIS returns several documents ranked by the number of hits, obtained by simply counting the number of times a search word appears in a document. To display a document, the user clicks on its title listed in the “Resulting Documents” field and a new window pops up with the content of the file.

Now, suppose the user found a document that is close to what s/he was looking for, but it was not exactly the right one: s/he can further refine the search by selecting the document and dragging it into the field marked “Similar to these Documents” on the WAIS Question panel, then reexecute the search.

5.3 Information Provider’s View

The information provider’s main job is to set up a public WAIS source, using the “`waissindex`” indexing routine and the “`waissserver`” network server routine. Both of these routines come with the WAIS software package for Unix. For more details about how to create an index, see Appendix C.

Data indexed by WAIS are stored in a single, read-only server and are not duplicated elsewhere, which simplifies the information provider's work and avoids inconsistencies. Moreover, data do not need to be structured in fields to be accessed by remote users and do not have to be updated periodically as for Archie, which needs to harvest the anonymous FTP servers monthly to build its composite index (see Table 5.1).

The reasons for co-locating the index and the resource data are twofold:

1. The indexing mechanism requires access to the entire content of the resource documents, as opposed to just the filenames and separate descriptions associated with them as for Archie;
2. As a consequence of the fact that the index size is about twice as large as the resource data, when there is a large amount of data distributed among many machines, exhaustive search is just not feasible. Only a restricted subset of the information space may be selectively searched, as Archie does, for example.

The top-level index, called *directory-of-servers*, registers and maintains a database of network addresses and descriptions of all publicly available WAIS servers. A directory entry gives enough information to connect to a WAIS server, including:

- the server IP address;
- the name of the database;
- the port number to connect at;
- the cost of information;
- contact information: maintainers;
- description of the server content.

It might also contain other auxiliary information, such as related servers, subscription information, user comments, etc. Cost and availability information can help guide the WAIS client application to alert its user to new choices of databases. If another related server is cheaper than the current server, then it could be suggested as an alternative server.

INFO.src — //textDeveloper/Examples/WAIS/sources

Access Information


Server: quake.think.com
 Service: 210
 Cost: 0
 Units: free
 Maintainer: wais-directory-of-servers@quake.think.com

[Send Mail to Maintainer](#)

Database Information

Database: INFO

Description: (This is exactly the same as the directory-of-servers source)
 This is a White Pages listing of WAIS servers. For more information on WAIS, use the WAIS system on the wais-docs server, or add yourself to the wais-discussion@think.com mailing list, or get the newest software from think.com/public/wais.

 To server makers: Please make new servers of text, pictures, music whatever. We will try to list all servers in the directory that get sent in to: directory-of-servers@quake.think.com (use the -register option on the command waisindex), but I reserve the right to take servers out if they are not consistently available. I will send notice to the maintainer before doing so.

CREATE

Part of the work of a system manager at a site which runs a WAIS server, once the local index has been created and made publicly available, is to enter by hand the description of the source and to register the server with the *directory-of-servers* at quake.think.com (see Appendix C).

Some commercial information products, such as the Dow Jones Information Service, provide their product through a WAIS interface. Companies that offer servers can make money by billing users directly, using credit cards, or by using 900 numbers to have the phone system bill the users. Direct billing is difficult to set up and can be expensive to operate, but large information providers might want to do this. Credit card billing has been popular, enabling any network to connect the user to the server and then the user is charged for use of the server. Typically, the first transaction with a server is a negotiation of how payment will occur and the allocation of a password for future transactions.

In November 1993 there were some 500 registered WAIS databases with an estimated 2000 additional databases that were not yet registered. There were approximately another 100 commercial WAIS databases. WAIS servers can be also accessed via transparent gateways by some Gopher and WWW clients.

Current work is attempting to add Boolean expressions and proximity and field specifications to queries.

CHAPTER 6

Gopher

The Gopher [2] project was first developed by Paul Lindner at the University of Minnesota to provide a simple, yet powerful, browsing system for distributed information.

6.1 Overview

Gopher is an RDS which allows users remotely to browse information stored on Internet hosts, known as *Gopher servers*. Since the filesystem organization is familiar to many users and it can be expressed in a simple syntax, Gopher represents its information space as a hierarchy of directories and files, where an item may be a pointer to documents residing on other Gopher servers.

The Gopher service architecture follows the client-server model. The Gopher protocol is designed to act primarily as a distributed document delivery system. The client software connects to a server and sends the server a selector (a line of text that may be empty) via a TCP connection at a preconfigured port, by default port number 70. The server responds with a block of text and closes the connection, as shown in the following example:

```
0About internet Gopher Stuff>About us rawBits.micro.umn.edu 70
1Microcomputer News&Prices Prices/ pserver.bookstore.umn.edu 70
```

Figure 6.1 shows an example which illustrates the components of a Gopher server reply, where each item is identified by:


```

0About internet Gopher Stuff:About us rawBits.micro.umn.edu 70
↑           ↑           ↑           ↑           ↑
type       user-visible name opaque selector hostname port

```

Figure 6.1: Components of a Gopher server response

type: the kind of object the item is;

user-visible name: a display string to show the item to the user in the menu listing;

opaque selector string: typically containing a pathname used by the server to locate the desired object;

hostname: which host to contact to obtain this item;

port number: the port at which the server process listens for connections.

The user only sees the user-visible name: in the example considered before, the first line describes a document the user will see as being entitled “About internet Gopher”. The Gopher client decides which items are available by looking at the first character of each line in a directory listing, and it can locate and retrieve any item by the triplet of selector, hostname, and port. To retrieve the document “About internet Gopher” for instance, the client software must send the retrieval string “Stuff:About us” to rawBits.micro.umn.edu at port 70. If the client does this, the server will respond with the contents of the document.

The first character of each line defines the type of item described on that line. In the example considered in Figure 6.1, the Gopher server returned:

- a directory list, identified by the first character on the line being “0”;
- a text document, identified by the first character on the line being “1”.

By giving a *type* attribute, it is possible to accommodate documents other than simple text documents. Gopher supports a diverse range of data such as video and audio format, allowing, for example, the display of selected types of image files and the play of sound files. In nearly every case, the Gopher client software gives the users an idea about what type of item this is by displaying an icon, a tag or the like.

The official Gopher types, from the Gopher Protocol Document [32], are shown in Table 6.1.

Code	Item definition	User display symbols
0	file	.
1	directory	/
2	phonebook server	<CSO>
3	Error	Error
4	BinHexed MacIntosh file	<HGX>
5	DOS binary archive	<PC Bin>
6	Unix uuencoded file	<), <Picture>
7	Index search server	<?>
8	pointer to a telnet session	<TEL>
9	binary file	<Bin>
+	redundant server (same as the previous server)	server name

Table 6.1: Gopher data types

The basic set of item types is *file* (character “0”), *directory* (character “1”) and *error* (character “3”). If a server-supplied directory listing marks a certain item with type character “2”, then it means that to use this item, the client should be able to employ the CSO (phonebook service) nameserver protocol. If a client does not understand what type an item is, then it simply ignores the item in the directory listing; the user never even sees it.

As the Gopher space escalated quickly, it became necessary to have an easy way to find Gopher items without doing a menu-by-menu, site-by-site search, and this motivated the development of Veronica.

Veronica [Very Easy Rodent-Oriented Net-wide Index to Computerized Archives] was designed by Steve Foster and Fred Barrie at University of Nevada [33] “as a response to the problem of resource discovery in the rapidly-expanding Gopher web. Frustrated comments in the net news-groups have recently reflected the need for such a service. Additional motivation came from the comments of naive Gopher users, several of whom assumed that a simple service would provide a means to find resources without having to know where they are” [34].

Veronica maintains an index of titles of Gopher items, and provides keyword searches on those titles. A Veronica search originates with a user's request for a search, submitted via a Gopher client. The result of a Veronica search is a set of Gopher-type data items, which is returned to the Gopher client in the form of a Gopher menu. The user can access any of the resultant data items by selecting it from the returned menu.

According to Table 3.1, Veronica is an Indexing Service, but since there are no Veronica clients "per se" and its servers can only be accessed through Gopher clients [35], it is discussed here. Because of Veronica, Gopher became a search-and-retrieval system as well as a browsing system.

A Veronica server typically offers a keyword search of hundreds of Gopher-server menus in the Gopher web. For the Gopher information space, Veronica is analogous to what Archie is for the anonymous FTP space, but unlike Archie, the results of a Veronica search can connect the Gopher client directly to the data source.

"Notice that these are NOT full-text searches of data at Gopher-server sites, just as Archie does not index the contents of ftp sites, but only the names of files at those sites: Veronica indexes the *titles* on all levels of the menus, for most Gopher sites in the Internet" [34].

Currently there are eight publicly-accessible Veronica servers, as illustrated by the items 2-9 in the Gopher menu "Search the World using Veronica" on page 44.

The Veronica service comprises two functions:

1. *Harvesting menu data* from Gopher servers periodically, and preparing it for use. The database is updated every week or in the worst case every two weeks [35]. A problem can be the fact that the Veronica servers are being run by independent entities and hence there is no guarantee that all servers will have the same data set.
2. *Offering searches* of that database to Gopher clients. The result of a Veronica search is a set of Gopher-type data items, which is returned to the Gopher client in the form of a Gopher menu. There are at present two versions of the Veronica search-engine. One version uses the NeXT's Digital Librarian and accepts only a single word to search for. The second version uses the WAIS index and it is preferred for three reasons:

- (a) it runs on more platforms;

- (b) it runs faster;
- (c) it allows partial Boolean searches.

These two functions are not necessarily provided by the same host computer. Most users and administrators of Veronica search servers will not need to be concerned with the first phase of the process. Operators of Veronica query-engines can obtain a prepared dataset for use with the query server.

6.2 User's View

Users can access Gopher in two ways:

1. through a **Gopher client**;
2. by **telnet** to a Gopher server.

In the latter case, from the network traffic point of view it is important to access a server that is located nearby. Table 6.2 shows some Gopher servers and the suggested area for use.

Gopher Server	Suggested area for use
consultant.micro.umn.edu	North America
gopher.uiuc.edu	North America
panda.uiowa.edu	North America
info.anu.edu.au	Australia
gopher.chalmers.se	Europe

Table 6.2: Gopher servers and suggested area for use

The root of Gopher hierarchy resides on host **rawBits.micro.umn.edu**. This is the default directory retrieved by Gopher clients when they are first installed. Clients can also be configured with other entry points into the hierarchy. This distributes the load over different servers and prevents a single point of failure.

Users can navigate through the available information space, organized as a graph rather than a tree because it can have pointers back to the same host, and containing items stored as:

- files on the corresponding server,
- directories, which can be distributed across multiple servers.

The user selects an item on the basis of a *menu* and the client retrieves it, concealing to the user the fact that the information might reside on different hosts.

Navigation through the Gopher information space is made easier by a list of *bookmarks* that can be kept to selected items. In this way, users can “jump” directly to an item without having to traverse all the Gopher menus to get there.

Gopher clients can also retrieve objects from WAIS, Archie and anonymous FTP servers by means of *gateways* servers. Free Gopher clients for different platforms are available and can be fetched by anonymous FTP [36].

Example: The preferred Gopher interface is the curses-based Unix client, but for this example let us assume the worst case, in which the user does not have a Gopher client and uses `telnet` to connect to a Gopher server:

```
> telnet gopher-server.cwis.uci.edu
```

```
Internet Gopher Information Client v1.12S
```

```
UC Irvine Gopher Server
```

- ```
--> 1. About UCI's Campus Information Service.
 2. Search UCI's Institutional Menus <?>
 3. Search All Menus (Institutional and Otherwise) at UCI <?>
 4. More about UCI's Campus Information Service/
 5. The Campus/
 6. The Classroom/
 7. The Community/
 8. The Library/
 9. The Researcher/
 10. The World/
 11. Computing & Networking at UCI/
 12. Newsletters, Listservs, and ZOTMail at UCI/
 13. Departmental Information Sources/
 14. Individuals' Information Sources/
```

- 15. Virtual Reference Desk/
- 16. Under Construction/

Each item is tagged with special characters shown in Table 6.1 to make it easy for the user to predict which kind of item is. Text files are denoted by a dot at the end of the menu entry. Menu items which end in "<?>" denote a searchable index. Indexes are like directories, except that one will only see a selected part of the directory content, based on the results of a keyword search. The slashes at the ends of certain lines denote that the item is a directory.

Suppose the user selects the line "10. The World/". This appears to be a directory, and so the user expects to see contents of the directory upon the request to fetch it. The following lines illustrate the ensuing Gopher menu:

#### The World

- > 1. About The World. ?
- 2. All UC Information Servers/
- 3. Internet Assistance/ ..
- 4. NSF's list of U.S. Government Gopher Servers/
- 5. Other Gopher and Information Servers/
- 6. Phone books at other institutions/
- 7. Search "The World" Using Veronica/
- 8. USENET News - UCI only/

The user does not know or care that the items up for selection may reside on many different machines anywhere on the Internet. Now the user supposedly choses

"7. Search "The World" Using Veronica/":

#### Search "The World" Using Veronica

- > 1. How to compose veronica queries (from Nevada).
- 2. veronica server at SCS Nevada <?>
- 3. veronica server at UNINETT/U. of Bergen <?>
- 4. veronica server at U.Texas, Dallas <?>
- 5. veronica server at University of Pisa <?>
- 6. veronica server at University of Koeln <?>
- 7. veronica server at PSINet <?>
- 8. veronica server at NYSERNet <?>

9. veronica server at SUNET <?>
10. Search Gopher Directory Titles at SCS Nevada <?>
11. Search Gopher Directory Titles at UNINETT/U. of Bergen <?>
12. Search Gopher Directory Titles at U.Texas, Dallas <?>
13. Search Gopher Directory Titles at University of Pisa <?>
14. Search Gopher Directory Titles at University of Koeln <?>
15. Search Gopher Directory Titles at PSINet <?>
16. Search Gopher Directory Titles at NYSERNet <?>
17. Search Gopher Directory Titles at SUNET <?>
18. veronica FAQ (from Nevada).

After selecting the index "9. veronica server at SUNET <?>", one is prompted for a keyword:

```

+-----veronica server at SUNET-----+
| |
| Words to search for |
| |
| |
| |
| |
| |
| |
| |
| |
+-----+

```

The user cannot see which Veronica search-engine is running, so one cannot know for instance if Boolean extensions are applicable. Assuming the user types the word "orbix", Veronica will look for the Gopher titles that contain that keyword, and Gopher will build a custom menu based on the result of the Veronica search and display it to the user as shown below:

veronica server at SUNET: orbix

- ```
--> 1. alformed cmd links
      2. iona announces orbix(tm) - object request broker.
```

When selecting the second menu item in the example considered above, which denotes a text file (see Table 6.1), the corresponding document is retrieved and displayed. To set up a reference to this file, the user can add it to his personal bookmarks list by pressing "a". Later on, when the user wishes to read this file again, he simply presses "v" to view the list of all the bookmarks and then select this item from it.

If the user wants a permanent copy of the file, not just a reference to it, he should press “s” to save the current item into a filename on his system.

6.3 Information Provider’s View

Gopher pointers to menus do not have to be periodically updated, unless when they require additions or modifications. This fact has two consequences:

- System administrators are spared the effort of harvesting data,
- but, on the other hand, inconsistencies might appear, such as if an item was removed and the menu item pointing to it has not been updated.

Gopher servers also provide information about any selected item, such as the host the item reside on, the port number for access and the path to help locate the item at the destination, the space occupied by the file if this is the case, etc.

Gopher servers have customized links to other hosts, and therefore they are not equivalent, in the sense that their databases do not contain the same information, except for redundant servers that mirror the same collection of files. If a connection to a server fails, a Gopher client should be able to select a redundant server to improve reliability and also to spread the work load among servers.

Gopher servers can also access anonymous FTP, WAIS and WWW servers, pointing at their top-level directory from which they display the information represented as a menu or search forms. Gopher links which refer to non-Gopher protocols are represented directly as URLs of the underlying access method and are not represented as Gopher URLs.

Many Gopher servers restrict user-access on a maximum number basis because the number of sessions on the server equals the number of files or directories the user clicked on, thus overloading both the machine and the network. Free Gopher servers are available via anonymous FTP [36].

To register a new Gopher server, connect to a Gopher server, select the menu item called “Gopher Server Registration” and fill in the form displayed as it follows:


```
+-----Gopher Server Registration-----+
|
| Administrator Name
| Administrator E-Mail
| GopherSpace Name
| Hostname of Server
| Port of Server          70
| Optional Selector
| Geographic Location
| Gopher+ Server (yes/no)?
|
| [Switch Fields: TAB] [Cancel: ^G] [Erase: ^U] [Accept: Enter]
|
+-----+
```

CHAPTER 7

WWW

WWW [3] is a distributed browsing system developed by Tim Berners-Lee at the European Laboratory for Particle Physics in Geneva (CERN), based on the hypertext paradigm.

7.1 Overview

Hypertext is text with pointers to other text. It allows annotations on a text to be saved separately from the reference document, yet still be tightly bound to the referent.

Hypermedia is a superset of hypertext: it is any medium with pointers to other media. This means that browsers may not only display text files, but may also display images or play sounds.

A **link**, in the sense of hypermedia, has two ends called **anchors** and it is directed from the *referent*, which is the source of the link, to the target of the link, called *reference*.

There are two different strategies for information discovery in a hypertext system:

Linking: the ability to define, manage, display, traverse links between items of information;

Dynamic search: the ability of finding information not by following a predefined link, but by dynamic search on text strings.

WWW is a browsing system which merges the techniques of hypertext, information retrieval and wide-area networking. Tim Berners-Lee describes the WWW world as consisting of *documents* and *links* [37].

Documents in the WWW information space can be “virtual” in the sense that they do not have to exist as files: they can be replies generated by a server in response to a query or document name. Indexes are special documents which may be found by following a link, but they may also be searched.

The WWW information space contains documents in many formats. Those documents which are hypertext, real or virtual, contain links to portions within themselves or links to other documents, or fragments within documents, locally or remotely.

The WWW model gets over the incompatibilities of data format between service providers and reader by allowing negotiation of format between the server and the client and this supports extension into multimedia: on appropriate platforms it can intermix text and images in a displayed document, the Mosaic client for XWindow for example.

WWW clients can access data via HyperText Transfer Protocol (HTTP) which runs over TCP, is faster than FTP for document retrieval and it allows also index search [37]. The syntax of an HTTP address follows the format of a URL as introduced in Chapter 3:

```
http: // <hostname> [: <port> ] [/ <path> ]
```

where one has to provide the Internet address of a WWW server, the port number for that server and the path to the document. The default port number for WWW servers is port 80, assigned by the Internet Assigned Numbers Authority, IANA. The WWW naming schemes also include other services, such as FTP, WAIS and Gopher, for which appropriate URL are built. Schemes currently defined are shown in Table 7.1.

The format of a hypertext name consists of:

- the naming *scheme* to be used;
- a *name* in a format particular to that scheme;
- an optional *anchor* identifier within the document.

```
<scheme>://<hostname>[:<port>]/<path>/<filename>[#<anchor>]
```

```
|-----|
name
```

file or ftp	Access is provided to files, using whatever means the browser and/or gateways have to reach files on obscure machines.
news	Access is provided to news articles, and newsgroups, normally using the NNTP protocol.
http	Access is provided to any other information using the HTTP search and retrieve protocol. The internal addressing of the information system is mapped onto a WWW path.
telnet	Access is provided by an interactive telnet session. This is provided ONLY as an interface to other existing online systems which cannot or have not been mapped onto the WWW space.
gopher	Access is provided using the Gopher protocol.
wais	Access is provided using the WAIS adaptation of the Z39.50 protocol.

Table 7.1: Schemes currently defined in hypertext addresses

where `<hostname>` is the name of the server in Internet form and `<port>` is the port number allocated to the service. The suffix `#<anchor>` represents the anchor identifier and it allows us to refer to a particular fragment within a document. The `<filename>` is optional; if not specified, the target of the link is within the same document by default. If the address ends with a `/`, the browser retrieves the list of files in the specified directory and generates a page of virtual hypertext pointing to its contents.

If a given hypertext node is an index, then a search may be done on that index by suffixing the name of the index with a list of keywords, after a question mark:

```
http://<hostname>[:<port>]/<path>/<address_of_index>?<keywordlist>
```

In the keywordlist, multiple keywords are separated by plus signs (+). If the search is successful, the document returned will contain anchors leading to other documents which match the selection criteria. The search method, and the logical and lexical functions, weights, etc. applied to the keywords will depend on the index address: one index may allow a search on author-given keywords only, while another may be a full text search. Some examples of hypertext addresses as used in WWW information space are illustrated below:

Example 1

```
file://nnsf.net/rfc/rfc977.txt#protocol  
or, equivalent,  
ftp://nnsf.net/rfc/rfc977.txt#protocol
```

This is a fully-qualified name, referring to the document `rfc977.txt` in the filename space of the given anonymous FTP server `nnsf.net`, and an anchor `protocol` within it.

Example 2

```
http://info.cern.ch/pub/www/illustrations/
```

This address gives access to the directory `illustrations` residing on the server located at `info.cern.ch`, and returns the list of files under it in a hypertext format.

Example 3

`http://crnvmc.cern.ch/FIND?sgml+examples`

This HTTP address makes the server at `crnvmc.cern.ch` perform a search on the words “sgml” and “examples”.

Systems such as WAIS, which are not currently accessed by WWW servers *directly*, may be accessed through gateways, in which case the document address is encoded within the HTTP address of the document in the gateway. Browsers which do not have the ability to use certain protocols may be in principle configured automatically to use certain gateways for addressing schemes (see Section 3.4.3 about Gateways).

7.2 User's View

To access the web users run a WWW client which is mainly a browser program. The browser reads documents, and it can fetch documents from remote sources. All documents, whether real or virtual, look similar to the user. WWW sessions start by the display of a *home page*, the document that is first read whenever WWW is launched. Users can customize their home page to link to interesting information anywhere in the WWW space. To modify the home page, one has to edit the file called `default.html` and located usually in the `WWW/` directory.

The WWW space can be browsed in three ways:

- **by following links:** browsers let users deal with the links in a transparent way: users select a link, and they are presented with the text that is pointed to.
- **by searching an index:** executing a keyword search on an index. The result of such a search is another document containing links to the documents found, the analogue of a Gopher menu returned by a Veronica search.
- **by navigating** around. Typical functions provided by the Navigate command can be interpreted in a hypertext web as follows:

Home	Go to initial node, called "home page"
Back	Go to the node visited before this one in chronological order.
Next	Go to the next target node which becomes the current node.
Previous	Go to the precedent node, the source of the link.

Currently there are three main ways of finding data from scratch on the web:

1. by *subject*;
2. by *organization*;
3. by *server type*.

Each of them represents an index accessible from the CERN home page at <http://www.cern.ch:80/>, or they can be accessed directly by entering the URL <http://info.cern.ch:80/hypertext/WWW/LineMode/Defaults/default.html>.

To access the classification on subject, one has to follow the link "by subject" in the "Other subjects" field from the CERN home page, or enter directly its URL, <http://info.cern.ch:80/hypertext/DataSources/bySubject/Overview.html>, and from there follow the links set to pages treating a particular topic, such as aeronautics, chemistry, history, law, etc. The tree of links about organizations is simply a list of pointers to some important research organizations, such as University of North Carolina, University of Oslo, CERN, MIT (Massachusetts Institute of Technology), NSF (National Science Foundation), and others. The list of information by server type at URL <http://info.cern.ch:80/hypertext/DataSources/ByAccess.html> represents a tree of Internet services such as Anonymous FTP, Gopher, WAIS, and for each of them it maintains a list of servers. A WWW client has to be built in such a way as to understand the protocol specified in the URL naming scheme to access the indicated service. So far, Mosaic, which works for the MacIntosh and XWindows environments, is the most complete WWW client.

As they find interesting information, users can build their own personalized web of documents by creating new documents and new links to new documents (for annotation, comment, etc.) or to already existing documents (for establishing new connections). However, users are allowed to modify only documents for which they

have permission to write, having just to follow other existing links with read-only access.

Beside the ease of use, an important advantage of the hypertext systems is that *information does not need to reside on the local host*, so it takes no space on the disk. References to an information object can be made instead of keeping a copy.

The main problems with navigation in a hypertext web are:

disorientation: the tendency to lose one's sense of location and direction in a non-linear document; and

cognitive overhead: the additional effort and concentration necessary to maintain several tasks or trails at the same time.

In addition, editing new documents in HTML format poses some difficulties, as currently there are only few hypertext editors, one of the best being incorporated in the WWW client for NeXT. See Appendix F about the HTML elements.

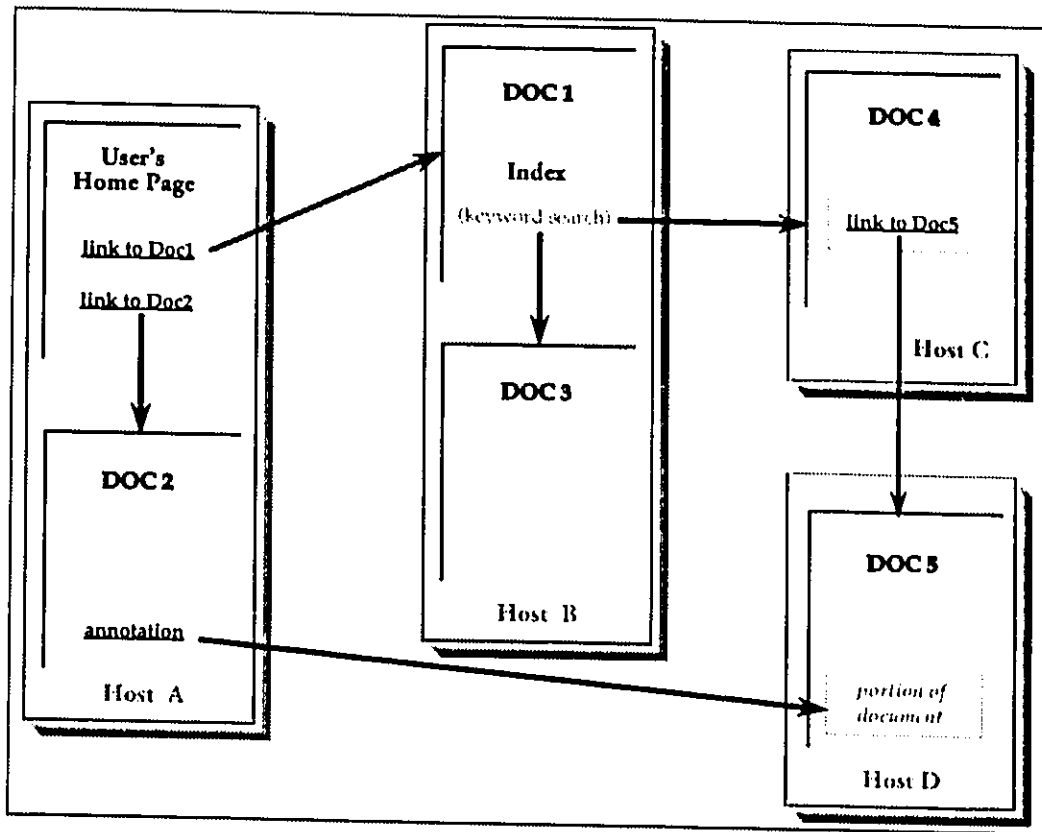
WWW allows users to annotate documents using hypertext links and this provides a basis for collaborative authoring. Users creating new documents become information providers if they agree to make them publicly available through one of the supported schemes: FTP, Gopher, WAIS, WWW. If the newly created document is in HTTP format, it is highly recommended to provide the author coordinates, name and address, for further comments from readers, as shown on page 56.

The variety of other supported protocols depends on the client software. For instance Mosaic, the WWW client for XWindows, can access all currently available schemes shown in Table 7.1. There is a range of free WWW clients, supporting many environments, for XWindows, MacIntosh, NeXT, etc., available through anonymous FTP [38].

To try the simplest WWW line mode browser, `telnet info.cern.ch`, no password required.

To try a full-screen browser, `telnet www.njit.edu` and login as "www".

Example: The best way to illustrate how WWW works is by seeing it in action, but for exemplification, consider the following picture:



In this example, the user starts from her customized home page and creates a link from there to a new local document, Doc 2. Then she follows an old link to a remote document which is actually an index, Doc 1. The user searches the index by typing a keyword, and as a result of the query, she is pointed to two other documents, Doc 3 and a certain portion within Doc 4. By clicking on the links, the documents are retrieved and displayed on the screen. The link to Doc 4 positions the display on the target portion of the document. Further, assume the user discards one of the documents, Doc 3, and decides to follow another link from Doc 4, which leads her to Doc 5. In Doc 5, the user supposedly found the information she had been looking for, and she creates an annotation by selecting a portion of text as a reference from her referent document, Doc 2. For the next sessions, whenever the user will click on the annotation she made in Doc 2, Doc 5 is fetched and displayed, with the appropriate part of the text being highlighted.

7.3 Information provider's view

Information providers set up hypermedia servers where browsers can get documents from. About how to set up a WWW server, see Appendix E. The server distributed as WWWDaemon includes two parts:

1. a common **HTDaemon** program which takes care of the communications;
2. a **HTRetrieve** function called by HTDaemon which only knows about finding files and sending them.

To make an index server, one needs an HTRetrieve that knows how to query a database or find information from keywords supplied after the “?” in the URL. Servers built as indexes return a tag called `<ISINDEX>` to notify the client that they accept searches. An example of a server built in this way is the WAISGate which speaks the WAIS protocol to contact a search engine.

The WWW system uses the HyperText Mark-up Language (HTML) format to represent hypertext documents for transmission over the network. Until recently, most hypertext documents were produced by hand. There are not many hyperdocuments available mainly because very few HTML editors exist and converters into HTML have just started to appear as the web spreads on the Internet.

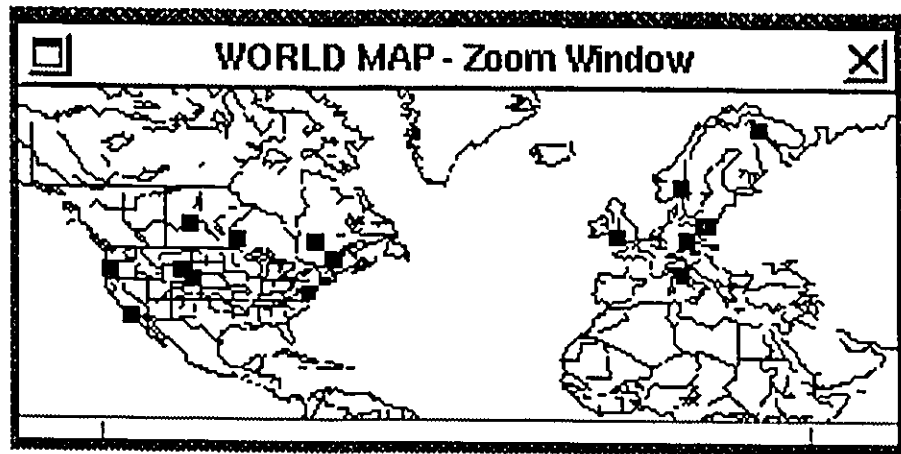
Data available through WWW do not have to be structured, but it is advisable to maintain on each host an overview document which people can use to get a quick idea (with pointers) of what information is available there. An important aspect of information which helps keep it up to date is that one can trace its author. To do this, one has to follow the sign-in steps indicated below:

- Make a signature-page for himself with his mail address and phone number;
- At the bottom of the file for which he is responsible, put a note with his initials or his name;
- Link the signature-page to that file.

As new servers show up, information providers must keep track of them and incorporate their data in the web by setting up new links, especially to an Overview file if there is one, from already existing documents that interested users might be reading or in an index which people might search. The links must be established in a logical

way, so that the web appears as a continuum to the users. The most common way of letting the system administrators know about the existence of a new server is so far to broadcast it on the Internet News.

In 1993 there were 24 WWW servers [39] accessible to WWW clients, most of them located in North America and Europe; as of February 1995 the number climbed over 1000 [40]. The picture below shows the display of WWW servers on the WIT world map¹.



¹Servers that have the same domain, eg. www.doc.ic.ac.uk, busby.leeds.ac.uk, www.ucl.ac.uk, etc. do not appear as separate buttons on the map.

World Information Tool

In this chapter we present the World Information Tool (WIT), a system which exploits existing Internet tools to provide a generalized Resource Discovery Service, based on a graphical representation of the distributed information discovery process.

8.1 Overview

WIT acts basically as a composite RDS, allowing a selection of tools and the interaction of their engines to search and browse items available in their information space. An important feature of this application is that users are able to query on type of service: anonymous FTP, WAIS [1], Gopher [2] and WWW [3], together with queries on subjects of interest.

WIT database contains data about information resources on the Internet, including their type and location, such that the application can select the appropriate protocol to be used in order to retrieve information objects and the corresponding client to view and to further exploit the individual resources. Furthermore, WIT considers the new approach of representing the information on a map analogue to hypertext, from which users can establish connections/links to any type of server displayed in the form of a “soft” button.

This application presents several relevant features:

- **Strategic choice of Internet tools** by bringing together the most popular RDSs to permit search, browsing, retrieval, editing, indexing, linking, and so on;

- **Automation of the resource discovery process** and consequently, a high degree of transparency to the user;
- **Manipulation of remote information** as if it was stored in the local filesystem;
- **Interaction of different RDSs engines** toward the refinement and further exploitation of the results, by creating some sort of *interdependency* between various systems;
- **No need to build or modify servers and protocols** because WIT is client-based.

8.2 Functionality

In the WIT system, the information is modelled as collections of typed objects, with each type featuring some attributes pertinent to one of the RDSs exploited by WIT. In particular, each object has associated with it at least an Internet host address and a geographical location, but it may have some additional attributes, such as a port number and a path.

Once located, resources are accessed via already existing access methods. An *access method* consists of the means of identifying and retrieving an object. Although the proposed architecture does not provide any additional access method, it does provide a way of determining all the information about an object which is available on the Internet. Moreover, the design of the WIT information engine does not require the development or the modification of any protocol. Its architecture is intended to be extensible, as the Internet expands, supporting in principle any search and access protocol.

According to the attributes gathered by querying an indexing service and the information stored in its databases, an appropriate and available access method is automatically chosen by WIT to retrieve the object.

WIT also provides a mechanism for offering information to users about Internet hosts represented as buttons on a world map, using either their IP addresses or their geographical coordinates.

For this purpose, the *Location Manager* (see Figure 8.1) was designed to allow users locate various types of servers on a map when they know only the host name

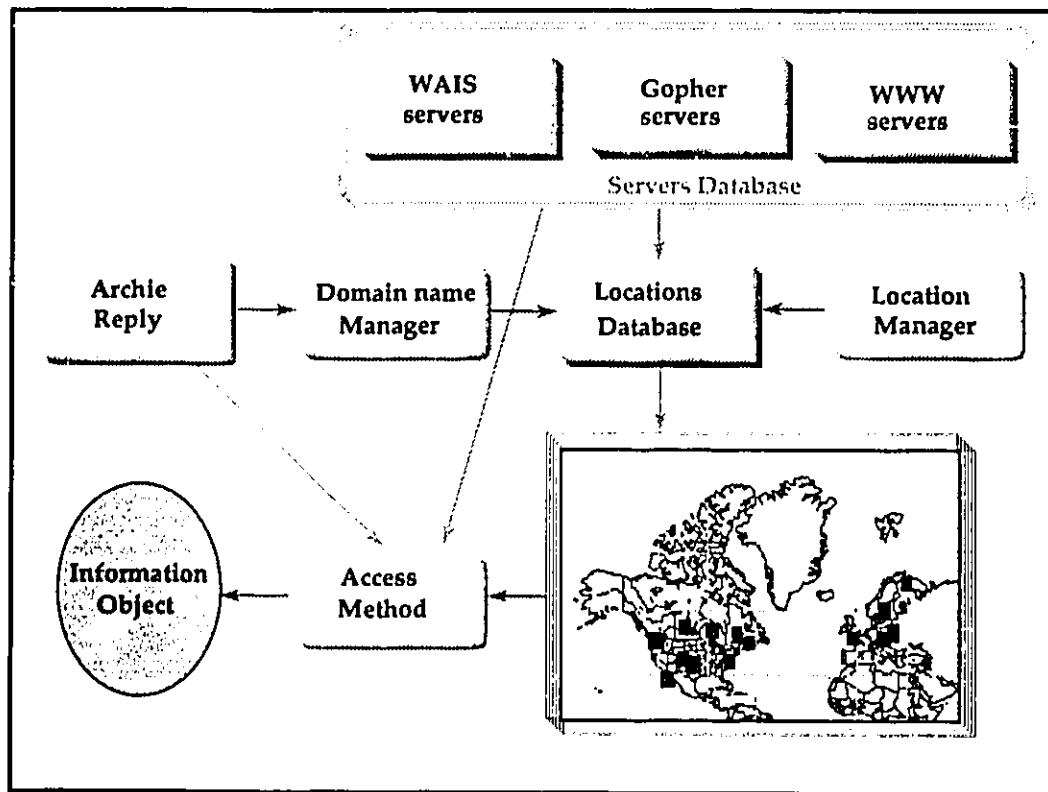


Figure 8.1: WIT information processing engine

and to permit users to find out the closest server within a certain distance, given only rough geographical coordinates.

The representation of various types of servers on the map as "soft" buttons permits a great flexibility and accords a better control to users over the network traffic.

And at last, but not in the least, continuous use of the hypertext capabilities facilitates the evolution of new threads of thinking.

8.3 Architecture

Resource discovery on the Internet implies two steps:

1. **Information Location;**
2. **Information Access.**

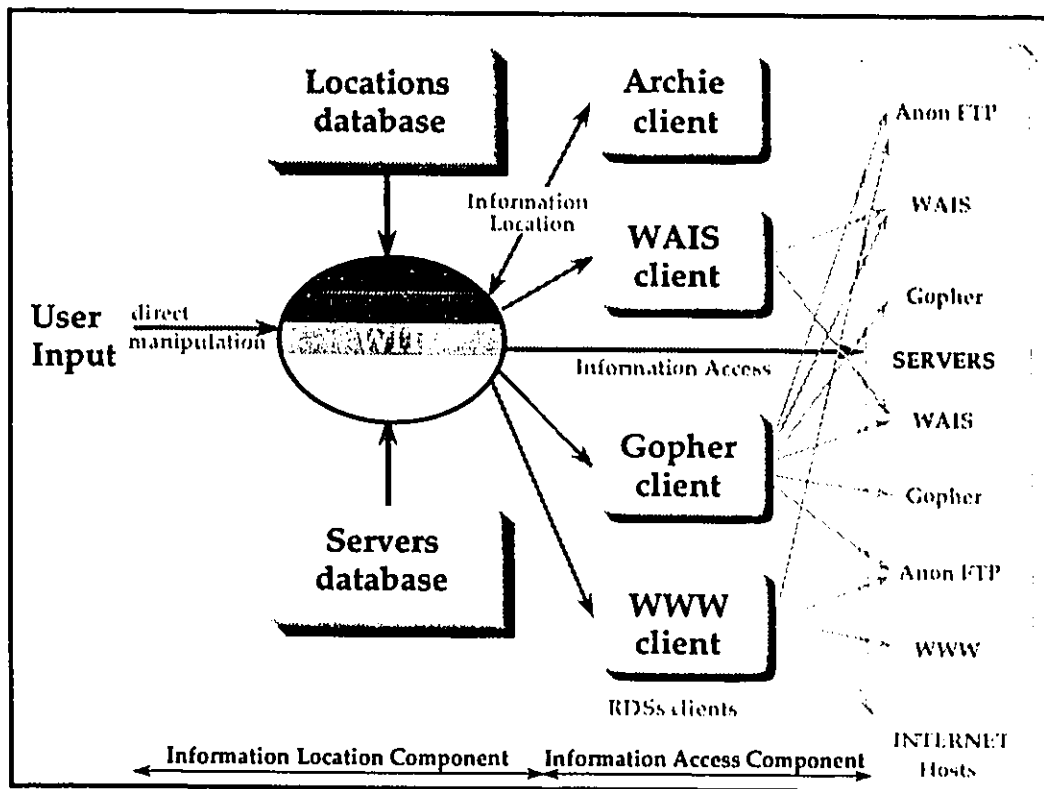


Figure 8.2: WIT Architecture

The similarity of the basic client-server model for RDSs makes possible the same partitioning of the WIT architecture: it is very modular and therefore open to the addition of new services, since it clearly separates the *location* and the *access* components of the system (see Figure 8.2).

The information necessary to the resource location is supplied by two large collections of data which may be searched separately:

1. the **Locations** database;
2. the **Servers** databases.

This feature of partitioning the collections of information allows WIT to scale as the system grows.

Information Location: WIT conveys the task of locating the information objects to Archie, as the default searching service, because by the large number of anonymous

FTP archive sites and the predominant weight the file transfer has among the Internet services (see Figure 2.1), it considers that most documents made public are available through anonymous FTP. Moreover, many Gopher, WWW and WAIS servers are anonymous FTP servers as well.

The result of the Archie query is returned to WIT and resolved into buttons on the map using the *Locations Database*, which contains entries of the form:

<domain_name> <geographical coordinates>

Since the number of Internet hosts is over 2 millions, we decided to match only the *domain* part of the Internet address of a host with its geographical location given the latitude and the longitude in degrees, minutes and seconds, as illustrated in the example bellow:

umanitoba.ca 49 54 39 N / 97 97 36 W

Currently there are 300 Internet domains represented in the Locations database. The resolution of the Internet address to the corresponding domain name of a host is performed by the *Domain Name Manager*. The mapping of a host name into the corresponding coordinates is processed both ways, its reverse operation allowing users to find out the name, the type and the exact location of a server given rough geographical coordinates, and for this WIT uses both the Locations and the Servers databases.

Information Access: Once the location of a resource has been established, one can proceed to its access. The mechanism for accessing information is represented by the access method associated with each type of service. The access method constructs a request for the appropriate server using parameters supplied by:

- the user input,
- the Servers databases,
- the result of a previous query.

For example, a resource identifier, returned in the Archie reply to a search for a keyword entered by the user, might refer to a specific file available through anonymous

FTP. In this case, the associated set of attributes necessary to fetch the file, would be the hostname, the path and the filename. Together, these supply enough information to access the file using the anonymous FTP protocol.

WIT refers automatically to the corresponding application of an underlying access method in order to view the instance of a resource, as follows: WAIS for Z39.50, Gopher for the gopher protocol, WWW for HTTP, and a particular remote browser and a window display built in WIT for the anonymous FTP.

Because RDSs make steady use of the network connections over the Internet when accessing the resources, where the information spaces overlap, as in the case of many Gopher servers which are at the same time anonymous FTP servers, users may try alternative services for information retrieval. This alternative is advisable also when there exist network problems which prevent connections, or the number of users allowed on a server reach its maximum limit.

As an example, the result of an Archie search might refer to an HTML file. In the case of Archie, the default access method is the one provided by the anonymous FTP service. The file is retrieved and displayed as a plain text file, but the advantages of hypertext are lost because the user cannot follow the links embedded in it. However, the HTML file can be retrieved by the WWW client instead, and in this way the user may navigate through the hypertext links, or she can use the WWW interface to view in HTML format the document retrieved by anonymous FTP, thus taking full advantage of the hypertext capabilities.

In order to permit the optimization of the use of the network bandwidth, WIT maintains a database of *Servers* which represents basically a collection of Internet host addresses. This database has three components, consisting of individual lists of all known servers for WAIS, Gopher and WWW. Archie does not provide an access protocol of its own, so it does not have a database of all the anonymous FTP servers, also because this would occupy disk space of the order of dozens of MBytes [41].

The Servers database has three basic roles in the functionality of WIT:

1. It helps users visualize the location of servers in order to optimize the network access and the workload of certain servers.
2. It provides information about the type of server selected by clicking on its "soft" button and the parameters necessary to the corresponding access method.
3. It allows users to formulate queries on the type of service.

A generic entry in each individual Servers database contains at least the Internet address of a server. In addition, for each RDS, the database maintains the values for the parameters required by the access method, as follows:

WAIS: An entry in the WAIS Servers database contains the port number allocated to the service and the source name on that server:

```
cnidr.org 210 directory-of-servers
<server_name> <port> <source_name>
```

This allows the access to a WAIS source for direct query, without having to pass by Archie. It is particularly useful when the location of a desired source is already known.

Gopher: It is enough to know the host name and to consider the default port number 70 to access a server and from its root, browse through the Gopher menus.

WWW: An entry in the WWW Servers database contains details referring to the "home page" at that site, which is accessed by default when the user clicks on the button corresponding to that WWW server:

```
www.unipi.it 0 /welcome.html 80
<server_name> <format> <path> <port>
```

The main advantage of this approach is that there are no single points of failure, no single server needing to be available since there are multiple ways of locating a resource. Therefore, the WIT system is expected to add to the robustness of the RDSs on which it operates.

Once the information is retrieved, WIT handles it according to its format: if it is a plain text file, then WIT displays a local copy in a separate scrolling window, while in the case of a directory name it brings up a remote browser. If the file is in HTML format for instance, then WIT passes the control over to WWW to allow a full use of its hypertext capabilities.

The other alternative is to send the user query through WIT directly to another indexing tool, WAIS for example, or to use Gopher or WWW, to browse the filesystem on a particular Internet host. In this case, the user may want to see first the distribution of the servers around the world, to choose the closest one among them.

WIT displays the existing servers as buttons on the map, lets users choose one, extracts the type of resource and then it selects the appropriate access method and the client to be used (see Figure 8.1).

A technique used by more experienced users is to *chain* queries from one engine to another to refine the search, because each RDS provides a more efficient use of some function.

8.4 Interface

Over the last year, the existence of WWW has prompted many institutions to set up their own servers and using creativity, they put up several maps with “lard” links in them. Users do not have any control over the setting of this kind of links which lead to documents chosen by the server administrator, so they can not be created or modified dynamically. The idea of “soft” buttons adopted by WIT from WWW is based on the same principle as the hypertext links which can be established by marking a certain portion of text in a document to refer to other remote documents.

Hence, the WIT system extends the hypertext paradigm introduced by WWW to traditional RDSs (Archie, WAIS, Gopher), using links as buttons on a map display to mark the discovery of information resources. WIT implements an interface of its own, yet fully compatible with the interfaces of the RDSs incorporated in it to pass control over to the RDSs clients when further need is.

The development of WIT started by using as a main window the North America map drawn by Gongyu Dai in MapView [42], a viewer for picture databases, and its seven possible projections:

1. Mercator (default);
2. Orthographic;
3. U.T.M.
4. GallMiller;
5. Albers;
6. Lambert Conformal;
7. Cylindrical Equal Area.

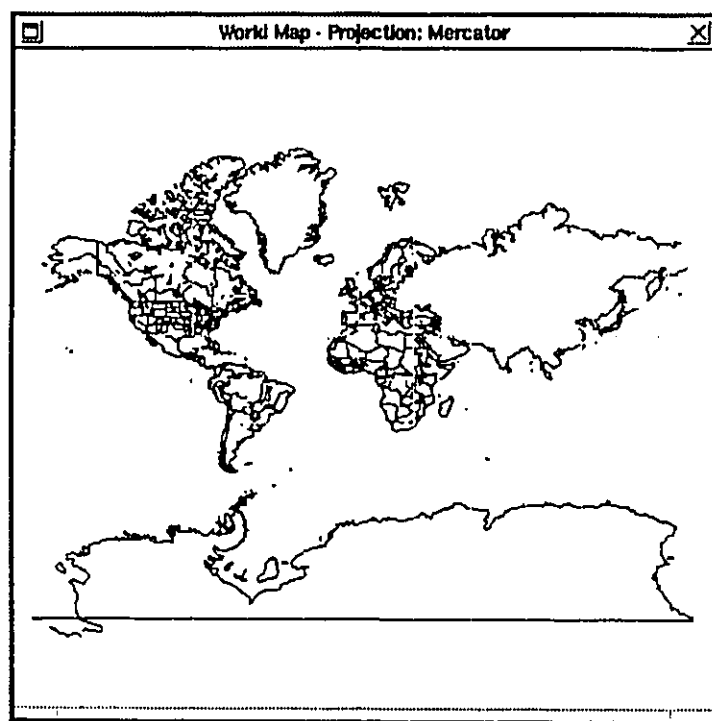


Figure 8.3: The World Map window

The name of the projection in which the map is represented, selected by the user, is indicated in the main window title. Gongyu Dai's application has been input the world map data, and slight modifications to extend the extreme coordinates to the world map and to show/hide the grid have been done. In the MapView application, the Orthographic projection was chosen as default for the North America continent, while WIT considers the Mercator projection as default, because it is more appropriate for displaying the world map.

Windows, buttons, panels and other graphic objects have the great advantage that they are a metaphor for the real world and they can be manipulated in the same way as familiar objects in the real world. WIT graphic user interface is intuitive and furthermore, it is consistent with other application interfaces, by using standard objects there where this is possible.

Users can directly manipulate the objects that appear on the screen, being able to control the whole application by simply choosing menu items, clicking on the "soft" buttons on the map, pressing buttons on panels or dragging the mouse to select a zoom view of a particular area on the map. Therefore, the users do not need to know the syntax of commands and type them to obtain the information needed. More experienced users have the facility of typing short keys, such as **Command-d** instead of clicking on the **Draw** menu item.

Moreover, where some features do not apply under certain circumstances, the corresponding menu items are disabled. For example, when WIT is first launched, only the valid menu items applicable at that specific time are written in bold characters, like **Projection**, **Draw**, **Hide**, **Quit**, all the others titles, **Internet**, **Grid**, being displayed in light grey to show those menu items cannot be selected.

8.5 Design Issues

WIT relies on existing protocols and already implemented clients to avoid the modification of the other services used by it. Therefore, the system *simulates* different RDS clients during the communication sessions with a server, by using the same access methods in a transparent way to the user.

For each RDS involved in the execution of WIT, the WIT application takes control over the RDS and sends commands to the appropriate server on behalf of the client, using the underlying access method. After retrieving the information, WIT chooses between the formats it recognizes, displays the information object and then it passes

the control to the corresponding RDS or leaves the user to choose one. In order to illustrate this procedure, we will consider some examples:

Archie: Suppose the user executes a search for the keyword “Internet”, using the Archie item from the submenu Internet/Tools. Since the Archie servers are supposed to keep the same information in their databases, the closest Archie server, `archie.cs.mcgill.ca` is used. The search is performed by default on the substring “Internet”, case insensitive and WIT issues the following command:

```
archie -s Internet
```

WIT collects and stores the Archie result in a temporary file which is both displayed to the user and used as an input for the Domain Name Manager, which extracts only the domain name from the Internet host addresses returned by Archie and matches them with the corresponding coordinates from the Locations database. The hosts mentioned in the Archie reply for which the domains were found in the Locations database are shown on the map as “soft” buttons. Lets suppose that among them, the user selects the one corresponding to the following entry in the Archie reply:

```
Host ajpo.sei.cmu.edu
```

```
Location: /public/resource
```

```
FILE -rw-r--r--      10753  Aug 15 12:26  internet.txt
```

When the user clicks on the button associated with the host `ajpo.sei.cmu.edu`, WIT chooses to use anonymous FTP as the most appropriate access method to retrieve the text file `internet.txt`. Therefore, the system initiates an FTP connection to the server `ajpo.sei.cmu.edu`, logging in as user “anonymous” and giving as password the login name of the user, changes the directory to `/public/resource` and gets the file `internet.txt` using the FTP command `get`, after which it displays the text in a separate window.

WAIS: WAIS servers are displayed on the map in the form of buttons, and after the user chooses one among them, WIT matches the server coordinates to find the host name of that machine, then it extracts from the WAIS Servers database the name of the source maintained at that site and from the arguments provided by this kind of information, it constructs a WAIS source structure with the name of that server and adds it to the list of sources kept locally:

```
(:source
  :version 3
  :ip-address "132.206.3.80"
  :ip-name "jimmy.cs.mcgill.ca"
  :tcp-port 210
  :database-name "example"
  :cost 0.00
  :cost-unit :free
  :maintainer "luminita@jimmy.cs.mcgill.ca"
  :subjects "WIT and WAIS"
  :description "This is an example of a WAIS source structure"
)
```

Then WIT automatically launches WAIS and the user may query the new source.

Gopher: The actions taken by WIT in the case of Gopher and WWW are similar because both services provide mainly browsing, such that the choice of a server implies starting the browsing from the root on that host. Thus, after selecting the Gopher menu item from the submenu Internet/Tools, and letting the user click on a button on the map, WIT connects to the Gopher server associated with that button at the default port number 70 and then passes the control over to the Gopher client, letting the user navigate through the menu items made available there. If the user wishes to view a particular file residing on a Gopher server, for which the URL is known, WIT builds a Gopher request from the arguments provided in the URL and uses `gophfilt`, a oneshot connection to a Gopher server, to fetch the document:

```
gophfilt -t0 -p "<path>/<filename>" -h <host> -s <port>
```

As a concrete example, consider the retrieval of the file called `Desc`, residing on the Gopher server `pogonip.scs.unr.edu`, in the directory `0/AboutGopher/Desc`. Instead of letting the user waste time browsing through the Gopher menus, WIT issues the following command:

```
gophfilt -t0 -p "0/AboutGopher/Desc" -h pogonip.scs.unr.edu -s 70
```

where the port number for the Gopher service is the default one, 70. The document is retrieved as plain text and then displayed by WIT in a scrolling window. If the user does not want to store locally the file available through Gopher, WIT keeps a pointer to it and adds it in the user's list of bookmarks, so that the user can find and retrieve later on the file, without wasting disk space. An example of the format of a Gopher bookmark built by WIT is the following:

```
#
Type=0
Name=Map Files
Path=0/Other-Internet-Resources/pictures/html-docs/mapfiles/test-maps.html
Host=gopher.lib.utk.edu
Port=70
```

where **Type** is the type of Gopher item as shown in Table 6.1, **Name** is the user visible name, as defined in Chapter 6, the **Path** indicates the path to the Gopher item, **Host** represents the Internet address of the Gopher server the object resides on, **Port** is the port number for the Gopher service. The sign **#** is used to separate different bookmarks in the list.

WWW: In the case the WWW service was chosen from the submenu Internet/Tools and the user clicked on a button displayed on the map with coordinates associated with a particular WWW server, WIT simulates a WWW client by connecting to that server at the port number indicated in the WWW Servers database and retrieving the home page at that site by extracting the path to it from the same database of servers:

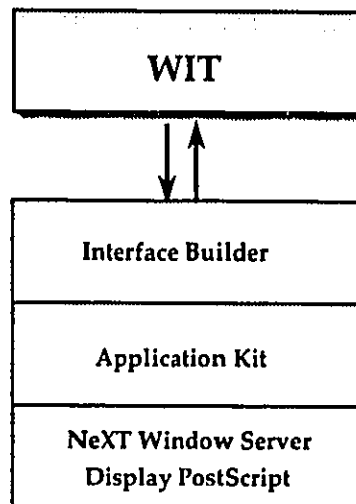
```
> telnet www.mcgill.ca 80
GET /
```


Naturally, a file in HTML format is retrieved via HTTP , then WIT launches WWW in order to display the document in hypertext format and to allow the navigation across the links.

8.6 Implementation

The WIT application was implemented using the NeXTStep tools:

- **the Interface Builder**, which provides access to a number of standard interface objects, such as panels, menu items, text fields, and also creates the user environment: places the windows, the panels, the menus on the screen. At running time, all these objects can be customized by direct manipulation: for example windows can be moved, miniaturized, resized, closed, and so on.
- **the Application Kit**, which consists of class definitions which can be used with the Interface Builder.
- **the Window Server**, which draws images using the NeXT implementation of the Display PostScript system.



New classes of objects have been created using the Objective-C language, such as MapView, Button and FTPBrowser. When possible, these classes have inherited the standard appearance, although their behaviour has been altered as necessary. User actions, are given through the mouse and the keyboard. The WIT main window keeps an event mask which determines the events that the Window Server can associate with

the window it manages. Each window maintains a *first responder* instance variable for the object that should handle the next event it receives. While running WIT, users can manipulate the mouse in two ways:

1. **clicking**, and
2. **dragging**.

Clicking is generally performed in order to select a menu item, press a button or choose a filename from the browser. Choosing a button on the map implies two steps, resulting in a double-click:

1. Click on the window to make it first responder;
2. Click on the button to select it.

Dragging is used to zoom in a particular area on the map, and the method has been implemented in Gongyu Dai's application to define a zoom perimeter.

The "network" part of the implementation has been done using Unix shell scripts to simulate different RDS clients. For example, to connect to a remote FTP site, the shell script receives the hostname and the user login name as parameters, together with the path and the filename to be retrieved and then it goes automatically through the whole FTP session, connecting there, changing the directory to the one indicated by the path and fetching the file specified in filename.

Separate methods have been implemented for each RDS, *wais*, *gopher*, *www*, as they need specific and a different number of parameters to be passed to the access method, as shown in Section 8.3 (see Figure 8.4).

The generic process of the application follows the main steps indicated below:

- draw world map;
- select Internet tools among: Archie, WAIS, Gopher, WWW;
- read the hostnames from the Servers database;
- extract their domain addresses from their Internet name;
- match every domain address with its geographical location stored in the Locations database (sequential search);

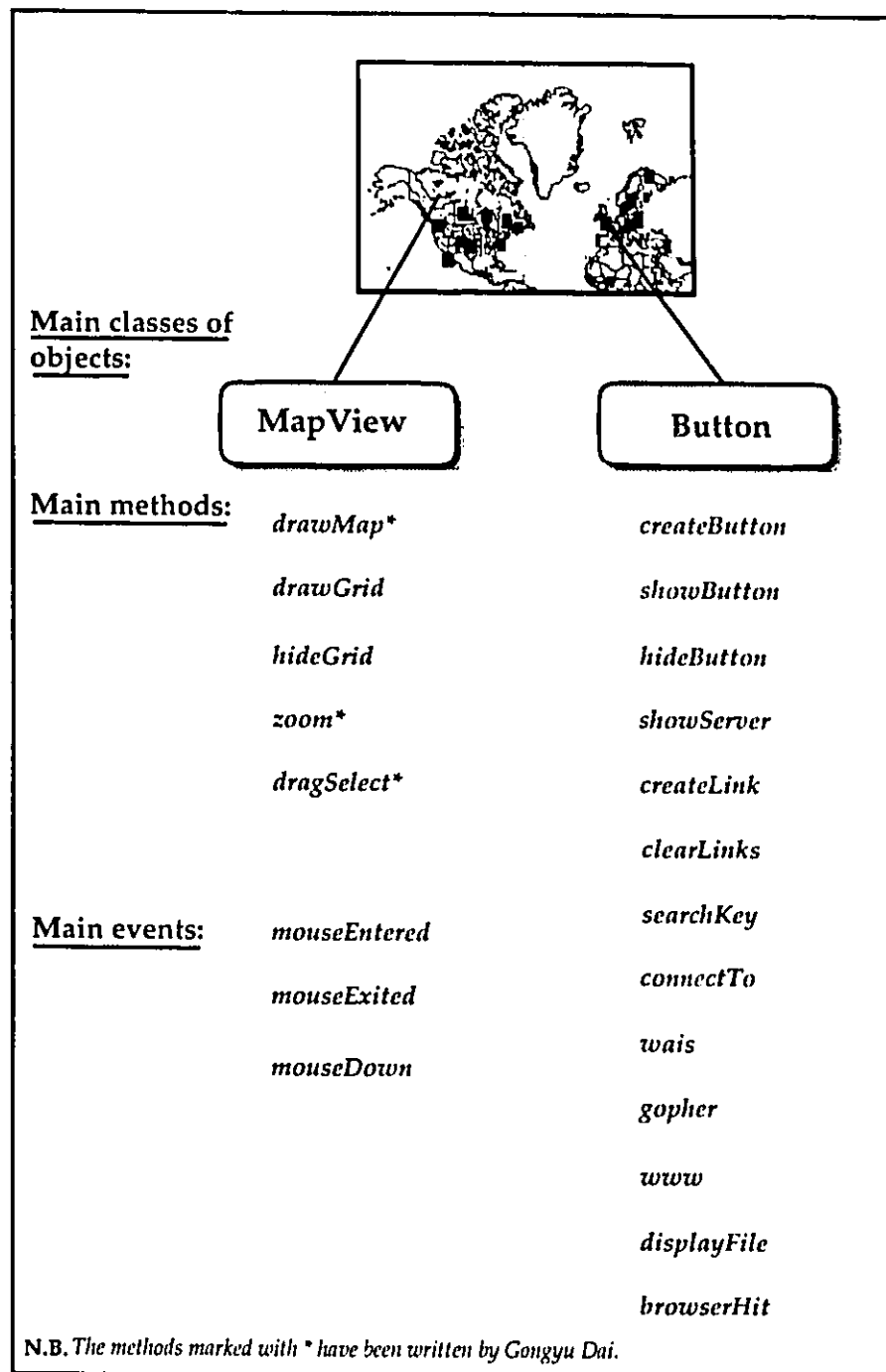
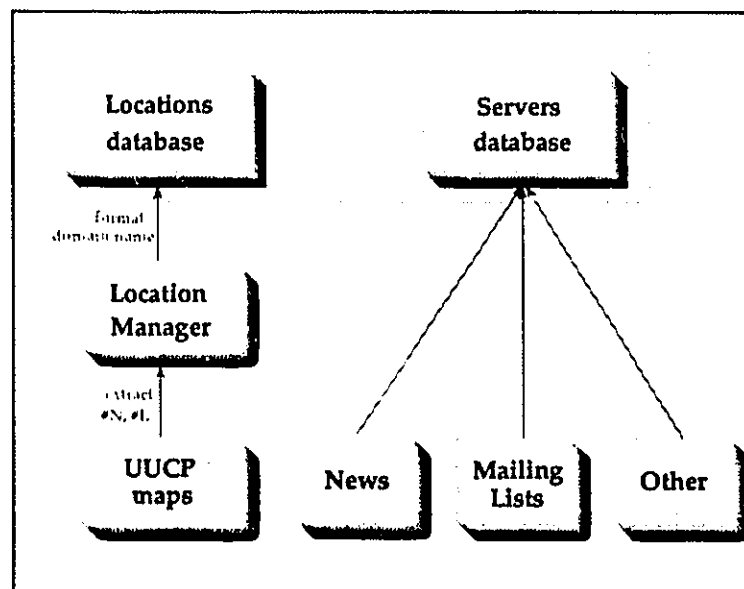


Figure 8.4: Main implementation issues

- display the appropriate hosts as buttons on the map;
- extract the corresponding information about the button selected by the user: server type, access method, parameters.
- establish connection to the server indicated by the button selected by the user;
- retrieve information object according to the type of server;
- display information object in a window if it is a file, or a browser, if it is a directory.
- launch the corresponding RDS, taking into account the format of the information object.

8.7 Maintenance

The information providers for each service are not required to modify their servers in any way. Given the diversity of ways in which new servers are advertised to the Internet world, it would be much easier if they registered with WIT, by providing their Internet host address and their geographical coordinates. The information location component may be easily updated with new entries using the data gathering program, but its structure should remain basically the same. The number of modules to be updated is restricted only to the Servers and Location databases.



The Location Database was built by extracting two fields from the UUCP maps [43], used to generate mail routes for those sites which generate email messages or netnews articles:

#N - Internet host name;

#L - Latitude and Longitude.

An example of a host entry in a UUCP map has the following format:

```
#N UUCP name of site
#S manufacturer machine model; operating system & version
#O organization name
#C contact person's name
#E contact person's electronic mail address
#T contact person's telephone number
#P organization's address
#L latitude / longitude
#R remarks
#U netnews neighbors
#W who last edited the entry ; date edited
```

For example, the host entry `eunet.de` located at 51 degrees, 29 minutes latitude North and 7 degrees, 25 minutes longitude East is represented as:

```
#N eunet.de
#S
#O EUnet Deutschland GmbH
#F
#E postmaster@Germany.EU.net
#T +49 231 972 2222
#P Emil-Figge-Str. 80, D-44227 Dortmund, Germany
#L 51 29 N / 07 25 E
#R EUnet Backbone Germany
#W jj@Germany.EU.net; 931006
```

Only the domain part of the Internet address is meaningful since we decided to locate hosts by their domain name. Since different countries use slightly different formats

for its entries, the data gathering program, written in *awk* (patter scanning and processing language), has been adjusted accordingly.

The UUCP mapping project is maintained by a group of volunteers and unfortunately the entries are not always updated. New entries are added when UUCP sites register with `domain-request@uunet.uu.net`. Corrections must be sent at `uucpmap@rutgers.UUCP`. For those sites not registered with the UUCP maps, we had to rely on the system administrators to provide us with the geographical coordinates of their site, or at least with their geographical location, city and country, where it was not possible to guess it from the Internet address of the host. Another alternative could have been to use the Geographic Location field required when for the registration of a Gopher server with the form illustrated in Chapter 6, but this field being optional, most administrators did not fill it.

New clients may be added as other access methods appear by simply selecting them according to the type of service provided. For each service, a new Servers database must be built, in order to allow the new RDS to interact with the other ones incorporated in WIT. The Servers database should contain all the information necessary to establish the parameters for the underlying access method.

CHAPTER 9

User's Manual

This chapter outlines the functional characteristics of WIT from the user's point of view, explaining the basic operations involved in using this application. Instructions for working with specific RDS clients were given in the previous chapters treating those services.

The WIT graphic interface built for NeXT represents a transparent way of automating the process of resource discovery by offering at the same time an easy-to-handle tool to users. Direct manipulation of objects on the screen allows even inexperienced users to customize the application interface to their needs.

Note: The examples offered in this project are done in general by necessity: they can not offer step by step directions that will fit every case. Furthermore, the Internet is constantly growing and changing, and new features to the RDSs incorporated in WIT are being made available on an almost monthly basis.

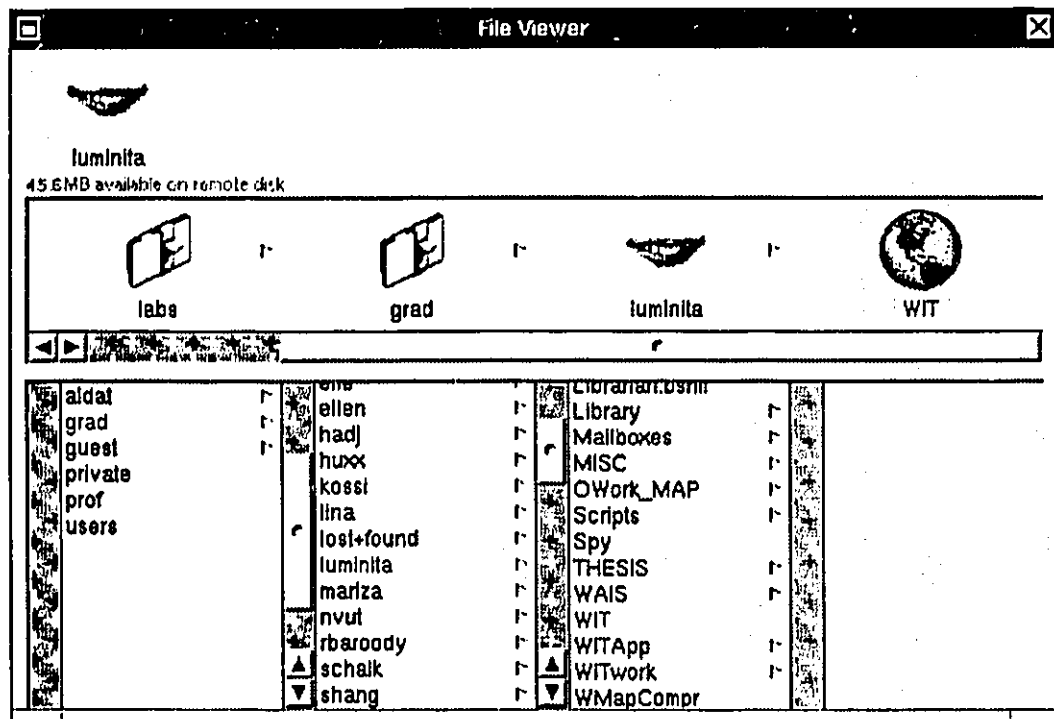
9.1 Getting started

This section explains how to start a work session with WIT on a NeXT station, once the application has been installed on the local host. Since Archie, Gopher and WWW, can be easily used with curses-based interfaces, there is no absolute need to get local client, access to a server being provided through `telnet` instead. It is advisable though to install and customize local client software because they give full advantage of the features offered by a service and have better time response.

In the examples illustrated in the present document we have used WWW and WAIS clients for NeXT and a Prospero-based client for Archie.

To launch the application, the following steps must be executed:

1. Select the directory where WIT has been installed from the FileViewer of the Workspace Manager;
2. Launch the WIT application by double clicking on its icon, illustrated below:



3. Click on the WorldMap window (shown in Figure 8.3) which pops up, to make it the active window;
4. Select "Projection" and then "Draw" from the main menu or, alternatively, press **Ctrl-d** from the keyboard, to start drawing the world map in the desired projection; by default, the map is drawn in the Mercator projection.

WorldMap	Projection	
Info...	Orthographic	o
Projection	Mercator	m
Draw	U. T. M.	u
Restore Ratio	GallMiller	G
Grid	Albers	a
Internet	Lambert Conf.	l
Hide	Cylin Eq Area	e
Quit		q

5. The "Grid" menu button is a toggle which can be switched ON/OFF, to display or to remove the grid on the map.

WIT guides the user along a session by enabling only the menu buttons associated with functions which are available at that moment and disabling the menu items corresponding to the rest. For instance, the buttons of the "Redraw", "Grid" and "Internet" are disabled right after the application is launched, because before taking any other action, the map must be drawn first.

9.2 Search strategies

WIT may initiate searches on three main groups of information, which can be carried on further by the RDSs incorporated in it:

1. **keyword**, querying Archie on both:

- filenames and
- descriptions of files.

2. **service**, selecting among:

- WAIS,
- Gopher,
- WWW.

3. **location**, specifying either:

- the Internet host name, or
- geographical coordinates.

In previous chapters we examined the RDSs relevant to the problem of resource discovery. Even if the information spaces of these services sometimes overlap, they do not coincide. One of the most important features of WIT is that the same query can be addressed to different types of service, thus broadening the information space covered by the search. Moreover, rather than expecting users to know the specific syntax to be used for each command, the initiative of operating a certain tool is taken by WIT, unless otherwise specified by the user.

Generic searches are performed using the Archie case insensitive substring search on a particular keyword entered by the user. For advanced users, other RDSs with search capabilities are available, according to the features shown in Table 3.1. For example, WAIS may be used to search on a keyword or on a topic. This alternative is particularly efficient when the user knows the type of service she requires (WAIS, in this case), the location of the server to choose on the map, or the name of the source.

9.2.1 Search on keyword

The search on a keyword is executed using the Archie service which indexes the anonymous FTP archives, the most probable sites where a public document might be found first.

Note: The following functions are tightly tied to the examples illustrated in a series of figures from the NeXT Interface. A general explanation of each step is followed by the specific actions.

Archie search

1. Start up the search by choosing the “Internet” menu item from the main menu, then “Tools” and finally the “Archie” submenu, which brings up the “Keyword Panel”.

WorldMap	Internet	Tools
Info...	Clear	Archie
Projection	Link	WWW
Draw	Show Links	Gopher
Restore Ratio	Tools	WAIS
Grid		
Internet		
Print		
Hide		
Quit		

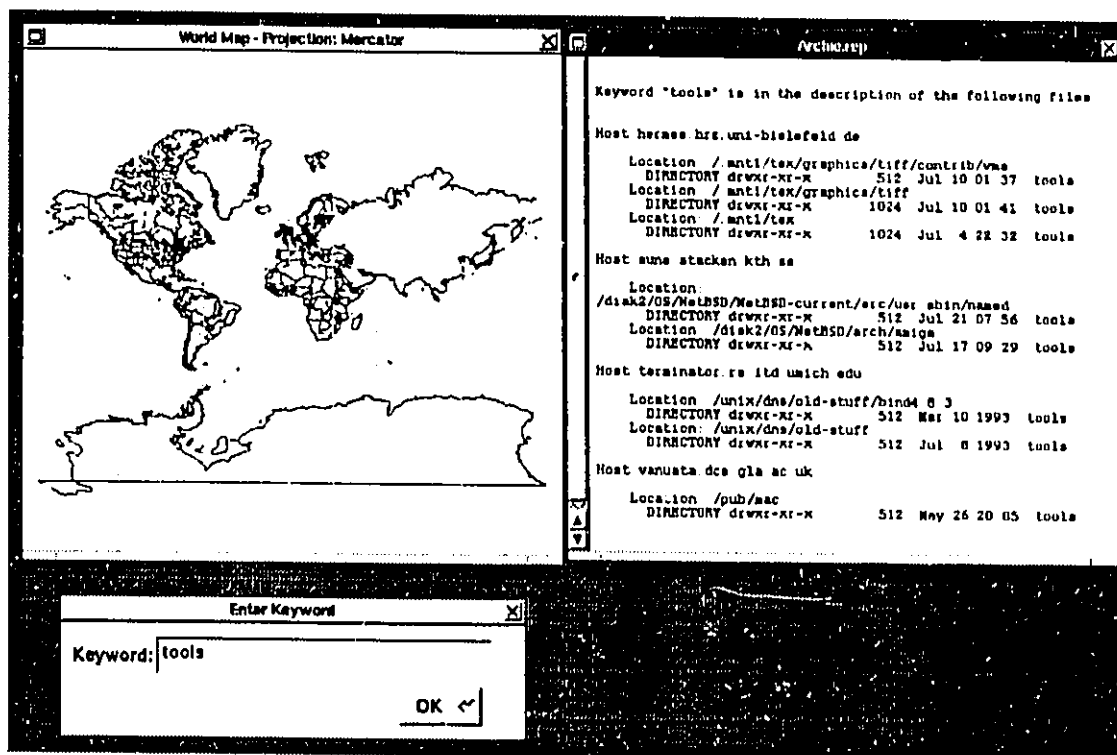
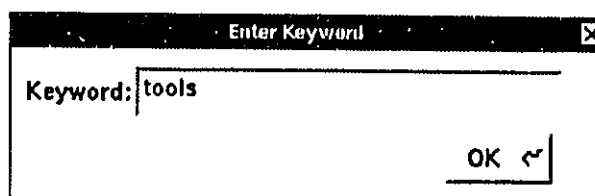


Figure 9.1: Archie reply to a query, as displayed by WIT

- Click on the "Keyword Panel" to make it active and type in a relevant keyword for the topic of interest, keeping in mind that the Archie search is executed by default on filenames and descriptions of files, for the substring indicated by the keyword.



If the query is successful, a new scrolling window containing the Archie reply to the search is displayed and the locations where the items found are stored appear as buttons on the map (see Figure 9.1).

- Choose a remote host by clicking on the corresponding button on the map: the "Server Connect Panel" containing the name of the server and its type (anonymous FTP, WAIS, Gopher or WWW) is brought up.

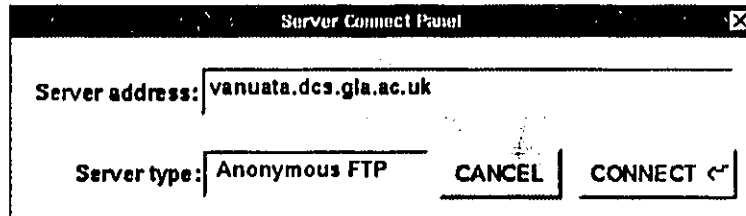
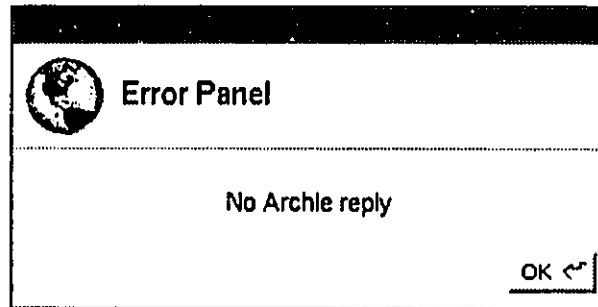


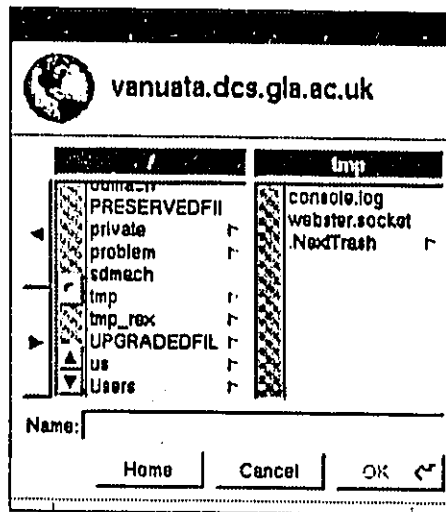
Figure 9.2: WIT Server Connect Panel

If the default Archie server cannot be reached, or the maximum number of users got to its limit and the server does not accept further connections, a "Error" panel appears, to alert the user about it.



To make the Error panel disappear, the user has to hit "OK". In this case, an alternate Archie server may be chosen or the user can select another service among WAIS, Gopher and WWW.

4. Establish a connection to the indicated site by pressing the "CONNECT" button on the "Server Connect Panel" or give up connecting there and in this case click on "CANCEL".
5. If the "CONNECT" button from the "Server Connect Panel" was selected, an Internet connection to the anonymous FTP server indicated on the panel is established. The information object located on that server is transparently retrieved by anonymous FTP and depending if it is a directory or a file, it is presented to the user in a browser or in a scrolling window.



The browser is entitled with the name of the remote host and it has the same appearance as a local browser, so that the user can use exactly the same commands he is accustomed with. When a file is selected from the remote browser, that file is retrieved by anonymous FTP and displayed in a separate window.

In the case of a HTML file, WIT retrieves it by anonymous FTP, but afterwards it launches WWW to display the document, in this way allowing the user to navigate across the hypertext links embedded in it.

9.2.2 Search on service

The search on the type of service covers the set of WAIS, Gopher and WWW, in an attempt to bridge the various ways in which these services are being accessed from the user's point of view. The interface designed for WIT makes transparent for the user the different methods of access for various RDSs, by displaying all servers as buttons on the map and automatically establishing connections to them.

First, the user chooses the desired service from the submenu of "Tools" item of the "Internet" main menu.

All the known servers available to that service are then displayed as buttons on the world map. The user selects one among them by clicking on its button and this action brings the "Server Connect Panel" up (see Figure 9.2). The name and the type of the chosen server are also shown in the panel. To connect there, the user presses on the CONNECT button.

WAIS: In this case, the name of the source residing on that server is extracted from the WAIS Servers database and it is included in the local list of WAIS sources, renamed with the Internet address of the server, so that after the WAIS application is automatically launched, the user can select the new source.

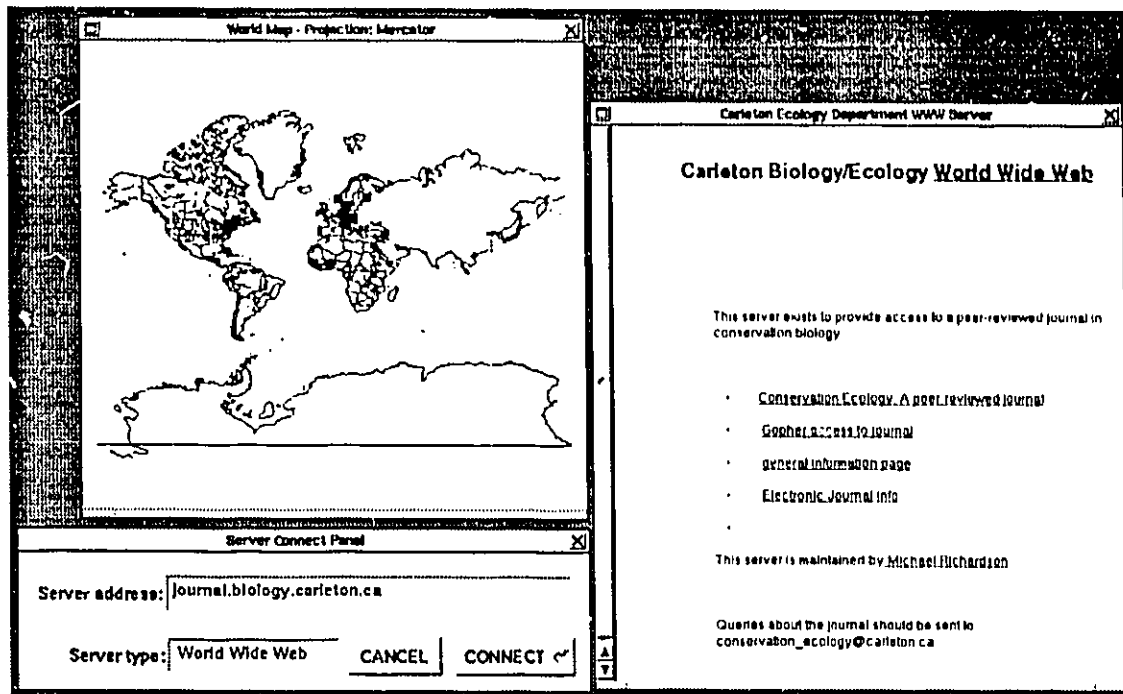
N.B. This approach of renaming the source has been considered in order not to destroy possible existing sources with the same name found in the list of the WAIS sources kept locally, when the WAIS Servers database of WIT is not updated, for instance.

Example:

- Suppose that after the display of the WAIS servers on the world map, the user clicks on the button associated with the WAIS server called `waiz.cpssc.ucalgary.ca`.
- The Server Connect Panel appears, to indicate the address of the Internet host and its type (in this case, it is both WAIS and Gopher server).
- The WAIS application is automatically launched and in the source list from the Source Palette appears one entitled `waiz.cpssc.ucalgary.ca.src` (see the following picture).
- By dragging this source from the Source Palette into the “*Using these sources*” slot on the WAIS Question window, a query may be made on a specific topic. Of course, other sources may be added as well, such that the search may be performed on several sources at the same time.
- Suppose the user types the following query in the “*Tell me about*” field on the WAIS Question window:

`information discovery`

The answer to this query is a list of documents shown in the “*Resulting Documents*” field on the WAIS Question window, in decreasing order of their matching scores:



The user can choose to follow the anchors embedded in the hypertext document in order to discover new information navigating through the web, to save the file for later use or to set up just a link to it from her own page.

9.2.3 Search on location

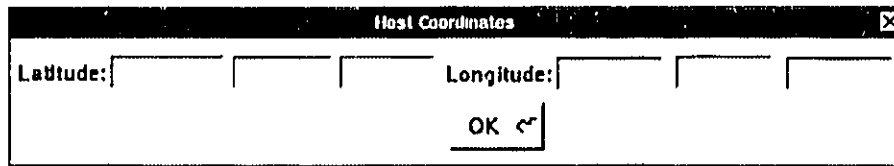
The search on location is performed both ways:

1. Find out the Internet address of the closest server within a specific distance, given rough geographical coordinates, and indicate it on the map.
2. Find out the geographical coordinates and implicitly locate on the map a server when only its Internet address is known;

Find the closest server: To get this function, the user has to select the "Coordinates" menu item from the submenu Internet/Link.

Internet	Link
Clear	Coordinates
Link	IP Address
Show Links	
Tools	

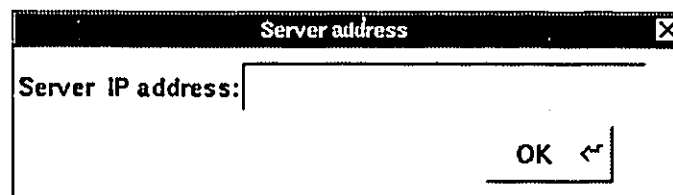
A panel with slots for the coordinates formatted in latitude and longitude, in degrees, minutes and seconds, is presented to the user.

A dialog box titled "Host Coordinates" with a close button (X) in the top right corner. It contains two input fields: "Latitude:" followed by three empty slots, and "Longitude:" followed by three empty slots. Below these fields is an "OK" button with a left arrow icon.

The user positions the cursor on each slot and types in it the corresponding values for the coordinates, then she presses the OK button on the panel to confirm the correct data entry. The closest server associated with those coordinates in the Locations database shows up as a button on the map.

Find the location when the IP address is known: This function is performed by an algorithm similar to the one that locates on the map all the known servers for a service, but in this case the method is slightly more complex because after matching the IP address of the server with the corresponding domain coordinates, the system has to look up the list of servers for each RDS and match the location with a type of service.

The user selects the "Link submenu from the "Internet" main menu item, then click on "sl IPAddress. This brings up a panel, entitled "Server address", where the user is invited to type in the Internet name of the server.

A dialog box titled "Server address" with a close button (X) in the top right corner. It contains a single input field labeled "Server IP address:". Below the field is an "OK" button with a left arrow icon.

9.3 RDS Interactions

Most of the time, users need several different functions to handle a certain task because where a service might offer an efficient search, for example, it may lack an indexing capability. Here WIT comes to help again, allowing users to exploit various tools at hand. The functionalities presented for different services can be chained in various sequences among themselves or together with utilities offered by other RDSs within WIT.

Example: Let us suppose an user would like to find documents regarding the problem of *"getting lost in the hyperspace"*.

- A generic search starts with Archie, by selecting it from the Internet/Tools submenu and typing in the Keyword Panel the word *"hyperspace"*.
- As a result, a list of anonymous FTP servers with files containing *"hyperspace"* is brought up in a scrolling window, together with buttons indicating where the hosts are located on the map:

```
ftp.pyramid.com containing the file exploring_hypertext.gz
unix.hensa.ac.uk containing the file exploring_hypertext.gz
bertha.pyramid.com containing the file exploring_hypertext.gz
```

- The user scans in turn the first two entries for the file *exploring_hypertext.gz*, but he gets the Error Panel informing her that the file does not exist there anymore (the database has been modified since the last update of Archie. For more details, see chapter 4 about Archie). From the third anonymous FTP site, *bertha.pyramid.com*, the file retrieved shows that its content is related to cyberpunk stories.
- At this point, the user might want to try a better "guess" and type *"navigation"*, the word she wants Archie to execute the new search on.
- Among the buttons displayed on the map as a result of the query, suppose the user selects the one associated with the server *xcf.berkeley.edu*. The file containing the substring *"navigation"* in its name or description is stored in the directory called *locat/pub/ht/projects/WWW/WWWBook/WWW/DesignIssues/Navigation.html*. WIT retrieves the file via anonymous FTP and since it is in HTML format, it launches WWW to display the document in hypertext format.
- *Navigation.html* contains a link to another hypertext document entitled *Navigational techniques and tools*, which in turns leads to other documents:

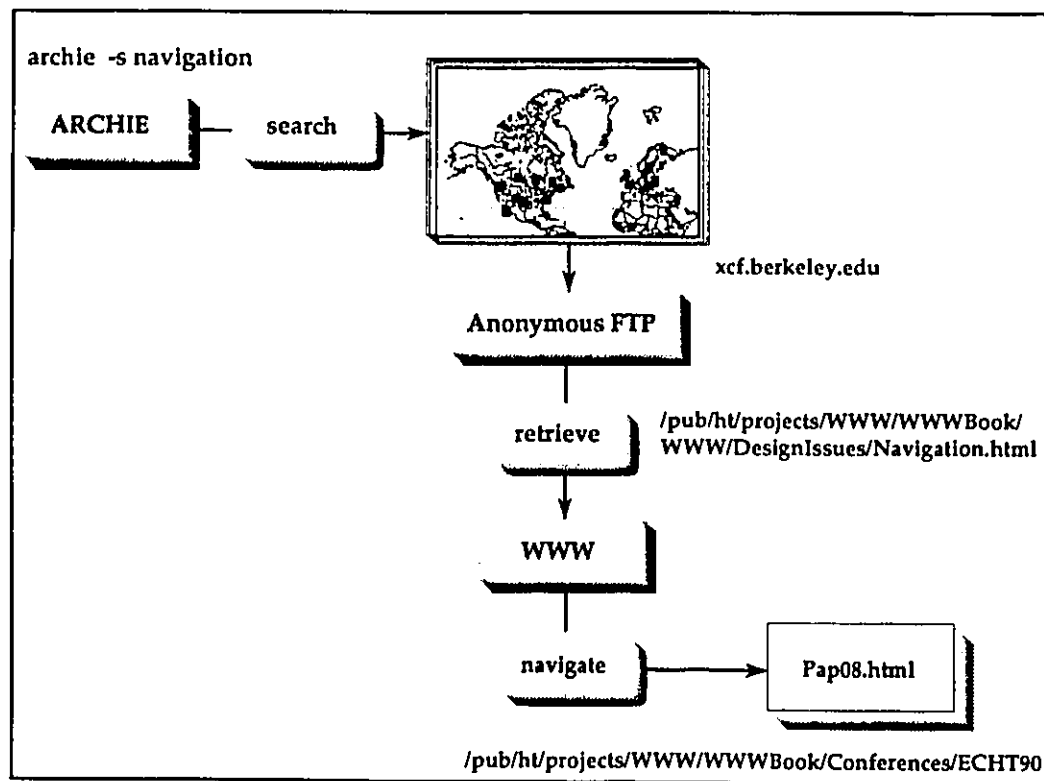
Defined structure;

Graphic Overview;

History mechanism;

Index.

- Now, the user can navigate across the links in the hypertext web, to discover new documents of interest. By doing so, in the directory /pub/ht/projects/WWW/WWWBook/Conferences/ECHT90 she finds in the file Pap08.html a reference to a document in which it is mentioned that in a paper presented at the first European Conference on HyperText (ECHT90), F. Biennier, M. Guivarch, J.M. Pinon stated: *"While browsing, users forget their goals..."* and this seems to be exactly the information the user was looking for.



- The user executes then an Archie search on the keyword ECHT90, getting the following results:

Host xcf.berkeley.edu

Location: /pub/ht/projects/WWW/WWWBook/Conferences/ECHT90

Host solaria.cc.gatech.edu

File: /pub/other/HCIbib/ ECHT90.bib.Z

Host archive.cis.ohio-state.edu
File: /pub/hcibib/ ECHT90.bib

The first one is the document she started the search from, so there is no need to retrieve the file one more time. The last two sites contain copies of the same documents kept by the HIC service. The HCI Bibliography is a free-access online extended bibliography on Human-Computer Interaction, that has reached 10000 entries with abstracts and/or tables of contents.

At archive.cis.ohio-state.edu she finally finds the bibliographic record of the paper she has been looking for:

%T Browsing in Hyperdocuments with the Assistance of a Neural
Network

%S Navigation and Browsing

%A Frederique Biennier %A Michel Guivarch %A Jean-Marie Pinon

%B Proceedings of the ECHT'90 European Conference on Hypertext

%D 1990

%P 288-297

%K Semantic browsing, Neural network, Dynamic path

%* (c) Copyright 1990 Cambridge University Press

%X The high degree of freedom a user has to browse through an hyperdocument often makes him puzzled. His main problems are first the expression of his informal need, sometimes using ideas associations, then finding the path in the hyperspace to reach relevant information. The guiding system proposed in this paper enriches the hyperdocument structure with undirect semantic links, i.e., associations between nodes according to their contents. Nodes' contents are connected to multimedia keywords called tags; direct and reverse associations between nodes and tags are embedded in a bidirectional neural network which allows inductive retrieval. One user controls the process thanks to some simple parameters: specialization level for selected nodes, precision of tags, inertia selector, tolerance functions for specialization and precision spreads. Upon request, the system dynamically raises a path that organizes the results of a query, contextual or not, adaptatively taking into account users' profile and special needs. Weights in the neural network may also be

slightly corrected from experience, adapting the association capability to users on their average.

- The user can then index the documents found, using WAIS as follows:
 1. Select File/New Source from the main menu.
 2. Hit CREATE button on the WAIS Source Panel which pops up. It asks for the name of the new source to be saved in, then it brings up the Create Database panel.
 3. The user is required to select the type of data he wishes to index from a pop-up menu entitled "Index As:".
 4. The user then has to drag the files or the folders he wishes to index from the FileViewer of the Workspace Manager into the slot next to the field called "Files and Folders o Index" on the Create Database panel.
 5. Press the INDEX button from the Create Database panel to give the new source indexed access to the information in the new database.

The index may be used later on for thorough searches on the documents which have been indexed for keywords, phrases, etc. It can also be completed with other documents of interest and made publicly available via a WAIS server (for details about how to set a WAIS server, see Appendix C).

CHAPTER 10

Conclusions

10.1 Contributions

WIT represents a valuable tool for the process of information discovery on the Internet because it uniformizes the use of the most popular RDSs it exploits and it provides access to many of the resources available through the Internet, allowing users to search and access different types of information from a single interface. The information can reside anywhere and on many different computer systems.

- WIT introduces a new graphical approach in exploiting RDSs, similar to hypertext, as it represents pointers to various types of information dynamically, resulting from user queries, as buttons on a map.
- WIT contributes to minimizing the network traffic by allowing users to have control over the choice of the host to connect to and showing them the closest hosts on an easy-to-understand map.
- WIT balances the workload among servers of the same type by offering alternative servers to the user, that is, Internet hosts which store the same information.
- WIT allows a high degree of processing hiding, automating the information discovery on the Internet transparently to the users, so that they do not need to be aware of different command syntax and hiding the variety of server types which are accessed.

- WIT enhances the role of distributed information on the Internet for users inexperienced with computers, providing a friendly, direct-manipulation interface.
- WIT easily scales up as new RDSs appear, because it is built at the client level of the client-server model of architecture, independently of the underlying protocol or server type.

WIT design also favors some security considerations; in particular, the user of such a system does not need to be granted specific access on the remote host to get the information being served.

It is easier to add new services to WIT, as they continue to appear, than setting new standards for the access methods. The problem of resource naming is still under discussion and although standardization holds great hope for the design and development of RDSs, there are still operational problems to be examined, related to the deployment of a single standard.

10.2 Further work

As the system will be frequently used, feedback is expected to improve its features, particularly for the functions needed to remove selected buttons from the map and to save others for later use.

For the time being, WIT main menu offers the possibility of deleting *all* buttons on the map in order to start a new query. To save a particular pointer to an information object or a whole file, WIT uses the Save, Link and Bookmark functions existent in the RDSs exploited by it, but presently there is no way to store selected buttons for later use.

A log might be kept as well, so users could navigate across different queries. For this, each query must have associated a list of buttons which resulted in response to the query, together with information pertinent to the access method to be used. Once this function is implemented, WIT will save disk space by storing only pointers to the files instead of keeping whole documents locally, independently of the service used: WAIS, Gopher or WWW.

Also crossed references to the answers returned by different services to the same query can be stored all together to allow interoperability of those services.

The data gathering component may be easily customized to get the IP address for a domain as well as the domain name, by using the Domain Name System (DNS),

designated to perform translations from the IP address to the domain name. This surely enlarges the database, taking more space, but it will allow the access by IP address as well as by domain name. For example, `mcgill.ca` and `3.80` refer to the same domain, so they can be kept together.

Recently, a new system called Alex [44] has been developed by Vincent Cate at Carnegie Mellon University to browse through anonymous FTP archives using the Network File System. It would be nice to exploit it in WIT.

The most efficient functions performed by each service may be then grouped together, to permit users to choose them rather than selecting a service. For example, the *Functions* menu may contain:

Search functions, provided by Archie, WAIS, Gopher;

Browse functions, provided by Gopher, WWW, Alex.

Finally, a way of discovering the availability of certain servers automatically, before trying to connect to them, may be envisaged to shorten the time of waiting for the connection to be established when it ends in a failure, maybe by using the Unix `ping`, designated for network testing, or `finger` commands. Adjacently, a way of aborting a connection might be implemented, as the time required for a response is sometimes long.

APPENDIX A

URL syntax in BNF

The mapping for some existing standard and experimental protocols is outlined in the BNF syntax definition.

url	httpaddress ftpaddress newsaddress nntpaddress prosperoaddress telnetaddress gopheraddress waisaddress mailtoaddress
httpaddress	http://hostport[/path][?search]
ftpaddress	ftp://login/path[!ftptype]
newsaddress	news:groupart
nntpaddress	nntp:group/digits
mailtoaddress	mailto:xalphas@hostname
waisaddress	waisindex waisdoc
waisindex	wais://hostport/database[?search]
waisdoc	wais://hostport/database/wtype/wpath
wpath	digits=path;[wpath]
groupart	* group article
group	ialpha[.group]
article	xalphas@host
database	xalphas

wtype	xalphas
prosperoaddress	prosperolink
prosperolink	prospero://hostport/hsoname[%00version[attributes]]
hsoname	path
version	digits
attributes	attribute[attributes]
attribute	alphanums
telnetaddress	telnet://login
gopheraddress	gopher://hostport[/gtype[selector]][?search]
login	[user[:password]@]hostport
hostport	host[:port]
host	hostname hostnumber
ftptype	A I D
formcode	N T C
cellname	hostname
hostname	ialpha[.hostname]
hostnumber	digits.digits.digits.digits
port	digits
selector	path
path	void segment[/path]
segment	xpalphas
search	xalphas[+search]
user	xalphas
password	xalphas
fragmentid	xalphas
gtype	xalpha
alphanum2	alpha digit - _ . +
xalpha	alpha digit safe extra escape
xalphas	xalpha [xalphas]
xpalpha	xalpha +
xpalphas	xpalpha[xpalpha]
ialpha	alpha[xalphas]

alpha	a b c d e f g h i j k l m n o p q r s t u v w x y z A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
digit	0 1 2 3 4 5 6 7 8 9
safe	\$ - _ \$\$. & + -
extra	" ' () , space
reserved	= ; / # ? :
escape	% hex hex
hex	digit a b c d e f A B C D E F
national	{ } vline [] ^ ~
punctuation	< >
digits	digit[digits]
alphanum	alpha digit
alphanums	alphanum[alphanums]
void	

APPENDIX B

URN syntax in BNF

urn	URN: Authority_Id : opaque_string
Authority_Id	Scheme_ID [: Individual]
Scheme_ID	IANA ISBN_Publisher_ID
Individual	xalphas
opaque_string	xalphas
xalphas	xalpha [xalphas]
xalpha	a b c d e f g h i j k l m n o p q r s t u v w x y z A B C D E F G H I J K L M N O P Q R S T U V W X Y Z 1 2 3 4 5 6 7 8 9 0 - _ . "

APPENDIX C

Setting up a WAIS server

- Select the data(files) to be indexed;
- Set up a directory with enough space to store the indexes and perhaps the source data. The source data cannot be moved or changed once the index is built, without re-building, since the index contains explicit references to the path and character positions in the file.
- Run **waisindex** on the data, directing the indexes into a disk area which has at least double size than the space occupied by the data.
- Add the following entry in the **/etc/services** file, that allocates the port number 210 to the z3950 protocol:

```
z3950    TCP    210
```

- Add the following lines to the **/etc/inetd** file to cause the **waissserver** routine to be called when a connection comes in on a z3950 port:

```
z3950 stream tcp nowait nobody \  
/usr/ccs/bin/waissserver waissserver.d \  
-d /usr/Local/lib/wais-data/public \  
-e /usr/spool/syslog/wais/wais-public
```

Note: Absolute paths must be used to specify file locations.

- Edit the <database>.src file to include a description of the new source. To register it for public access, send a copy of <database>.src to **wais-directory-of-servers@quake.think.com**.

APPENDIX D

Setting up a Gopher server

The Unix Gopher server, called `gopherd`, allows the publication of documents on the Internet using the Gopher protocol. A copy of the software may be obtained via anonymous FTP from the software repository at `boombox.micro.umn.edu` in the `/pub/gopher` directory. The source code requires about one megabyte of disk space when uncompressed and two megabytes of disk space when compiled.

Before the compilation, one has to set up properly the following parameters in the `Makefile.config` and `conf.h` files, both found at the top level of the Gopher source directory, as shown in Table D.1 and Table D.2.

- To make the server, type:

```
make server
```

- To install the server, type:

```
cd gopherd
make install
```

- Create with the command `mkdir` the Gopher-data directory, which will contain all the information that Gopher clients will see.
- One can add data to his own server by creating what are known as *Links* in the Gopher-data directory tree. Links allow servers to be connected together. To make a link, create a file starting with a period. A link file can contain multiple links, as shown in the following example:

Parameter	Default	Meaning
PREFIX	/usr/local	The directory in which the software will be installed
CLIENTDIR	(PREFIX)/bin	Where the Gopher client is installed
CLIENTLIB	(PREFIX)/lib	Where the help files for the client are
SERVERDIR	(PREFIX)/etc	Where the server files are installed
DOMAIN	hostname	The hostname of the Unix machine
SERVERDATA	/home/gopher	The default location of data for the Gopher server
SERVERPORT	70	The default port for the Gopher server
SERVEROPTS	SERVEROPTS	Add -DLOADRESTRICT to restrict access to the server at a certain load average and also add -lkvm to SERVERLIBS
DOMAIN	hostname	The hostname of the Unix machine

Table D.1: Configuration parameters in Makefile.config

```

Type=1
Name=Luminita link to VERONICA(Nevada)
Path=1/veronica
Host=futique.scs.unr.edu
Port=70
#
Type=1
Name=National Weather Service Forecasts
Path=1/Weather
Host=ashpool.micro.umn.edu
Port=70

```

The `Name=...` is what the user will see when navigating through the Gopher space. The `Type=...` defines what kind of item the object is (see Table 6.1). The `Path=...` is the selector string the client will use to retrieve the actual document. The `Host=...` and `Port=...` specify a fully qualified domain name and a port for an Internet host. An easy way to copy interesting links is to

Parameter	Default	Meaning
CLIENT1_HOST	<code>gopher.tc.umn.edu</code>	Prime server to connect to
CLIENT2_HOST	<code>gopher.tc2.umn.edu</code>	Alternate server to connect to
CLIENT1_PORT	70	Default port for the prime server
CLIENT2_PORT	70	Default port for the alternate server
PAGER_COMMAND	<code>more -d %s</code>	Unix command used to display text
PLAY_COMMAND	<code>soundtool</code>	Command used to play sound files
IMAGE_COMMAND	<code>xv</code>	Command used to view image files
MAIL_COMMAND	<code>mail</code>	Command used to mail files
TELNET_COMMAND	<code>telnet</code>	Connect to other hosts using telnet

Table D.2: Configuration parameters in `conf.h`

use the curses-based Gopher client for Unix. By pressing “=” one gets all the information suitable to be included in a Link file.

- Certain files can be ignored by modifying the `gopherd.conf` file. If one wants the server to ignore all files with the extension `.html`, then one must add the following line to the `gopherd.conf` file:

```
ignore: *.html
```

- The server may be started at the boot time by adding a typical entry for the `etc/rc.local` file in the file `/etc/start-gopher-server`, as in the following example:

```
if [ -f /home/Gopher-data ] then
  /usr/local/etc/gopherd -u nobody \
  -l /home/GopherLog70 Gopher-data 70
endif
```

- Reboot the machine and the Gopher server will start running continuously.
- One can run the Gopher server using the `inetd` instead of running it continuously. In this way, the server starts running only when a connection is made by a client. To do this, one has to modify two files:

1. `/etc/services`, by adding the following line:

```
gopher 70/tcp
```

2. `/etc/inetd.conf`, by adding a similar line:

```
gopher stream tcp nowait nobody /usr/local/etc/gopherd gopherd
```

- To start the Gopher server you should either reboot the machine, or send a HUP signal to the process id of the `inetd` process:

```
kill -HUP process_id
```

APPENDIX E

Setting up a WWW server

This is how to set up the Internet Daemon, *inetd*, to run *httpd* whenever a request comes in.

- Install *httpd* binary: copy *httpd* into a suitable directory, such as */usr/etc*. Make the root the owner of it and then *chmod 755 httpd* to make it writable for the root only.
- Add *httpd* to the */etc/services* file, writing a line like the following:

```
http 80/tcp # WWW server
```

NOTE:

The NeXT uses the *netinfo* database instead of the */etc/services* file. This is managed with the */NextAdmin/NetInfoManager* application. Here is how to add the *http* service:

1. Start the *NetInfomanager* by double-clicking on its icon.
2. If you are operating in a cluster, open either your local domain (hostname) or if you have authority, the whole cluster domain (/). If you are not in a cluster, just use the domain you are presented with.
3. Select *services* from the browser tree.
4. Select *ftp* from the list of services.
5. Select *duplicate* from the edit menu.

6. Select *copy of ftp* and double-click on its icon to get the property editor.
 7. Click on *name* and then on the value *copy of ftp*. Change this to *www* by typing *www* in the window at the bottom, and hitting return.
 8. Click on *port*, and then on the value *21*. Change it to *80*.
 9. Use *Directory:Save* menu (Command/s) to save the result. You must have root or netinfo manager privileges and password.
- Put the following line in the Internet Daemon configuration file, */etc/inetd.conf*:

```
http stream tcp nowait root /usr/etc/httpd httpd
```

- Send a HUP signal to *inetd*. After updating the configuration file of the Internet Daemon, *inetd.conf*, find out the process number under which *inetd* runs and send a *HUP* signal to it:

```
> ps -aux | grep inetd
root 123  0.0  1.2 1.24M 256K ?   SW   0:02 /usr/etc/inetd
> kill -HUP 123
```

- To test the server, simulate a client by:

```
telnet your.host.domain 80
GET /document
```

APPENDIX F

HTML elements

The HyperText Markup Language is defined in terms of the ISO Standard Generalized Markup Language (SGML). SGML is a system for defining structured document types and markup languages to represent instances of those document types.

An HTML instance is like a text file, except that some of the characters are interpreted as markup. The markup gives structure to the document.

The instance represents a hierarchy of elements. Each element has: a name, some attributes and some content.

Every element is represented in the document as a start tag, which gives the name and attributes, followed by the content, followed by the end tag.

Example of HTML instance

```
<HTML>
  <TITLE>
    A sample HTML instance
  </TITLE>
  <H1>
    An Example of Structure
  </H1>
  Here is a paragraph.
  <P>
  <UL>
    <LI>
```

```

        Item 1 has an <A NAME="anchor"> anchor </A>
    <LI>
        Here is item 2.
    </UL>
</HTML>

```

Start tags are delimited by < and >, and end tags are delimited by </ and >. In a start tag there might be present several *attributes*. An attribute consists of:

a name;

an equal sign;

a value.

Comments may be included in an HTML document by surrounding them with <!-- and -->.

Documents contain an initial **HEAD** element followed by a **BODY** element. The following elements which appear in a **HEAD** element define the properties of a document:

TITLE	The title of the document
ISINDEX	Sent by a server in a searchable document
NEXTID	A parameter used by editors to generate unique identifiers
LINK	Relationship between this document and another.

The following are elements which may occur within the **BODY** element of a document:

Headings	Several levels of heading are supported.
Anchors	Sections of text which form the beginning and/or end of hypertext links are called <i>anchors</i> and defined by the A tag.
Paragraph marks	The P element marks the break between two paragraphs.
Address style	An ADDRESS element is displayed in a particular style.

Blockquote style	A block of text quoted from another source.
Lists	Bulleted lists, glossaries, etc.
Preformatted text	Sections in fixed-width font for preformatted text.
Character highlighting	Formatting elements which do not cause paragraph breaks.
Graphics	The IMG tag allows inline graphics.

Attributes of the **anchor** tag are as follows:

- HREF** OPTIONAL. If the HREF attribute is present, the anchor is the *start* of a link. If the reader selects this text, (s)he is presented with another document whose address is defined by the value of the HREF attribute. The form **HREF=#<identifier>** refers to another anchor in the same document.
- NAME** OPTIONAL. If present, the attribute NAME allows the anchor to be the *destination* of a link. The value of the attribute is an identifier for the anchor. Identifiers are unique strings within the HTML document. Another document can then refer explicitly to this anchor by putting the identifier after the address, separated by a hash sign .

Example

For more details about WWW see the `CERN
` home page.

The **address** element, often found at the end of a document, is intended for information about the author of the document, signature, email address for collaboration purposes.

Example of Address:

```
<ADDRESS><A HREF=Author.html>L.C. Stancu</A></ADDRESS>
```

APPENDIX G

WAISStation User Manual

The model for a WAISStation search is:

- Start with a few key words or phrases (as usual).
- Examine the articles WAISStation retrieves.
- Indicate to WAISStation which articles, or which sections of articles, are most useful (i.e. closest to the subject). Ask it to search again, using those selections as models.
- Repeat the process until the user has found what she want.

Getting started: Double-click on the WAISStation icon: The SOURCES and QUESTIONS palettes will appear. Sources are generally available to everyone at a given site. Questions, on the other hand, generally belong to individuals.

Creating a new question:

1. From the QUESTION menu, select NEW QUESTION. A new QUESTION panel opens.
2. Choose a source that you want your Question to search. Drag it from the SOURCES palette into the “*From These Sources*” field in the QUESTION palette. The source name will appear in the sources field. Do this for as many sources as you wish.

3. Type words or phrases in the “*Tell me about*” text field, describing the subjects about which you want information. WAISStation searches for these words to find useful documents.
4. Click on the SEARCH button to run the question.

Getting results:

1. After the user hits the SEARCH button on the WAIS Question window, the cursor will turn while WAISStation searches for documents which match the request. The “*Resulting Documents*” field then displays the titles of the most useful documents found. Documents are ranked from 0 to 3 stars, depending on how well they match the question: the best matches are at the top of the list. Drag down the scroll bar to see more documents.

2. Double-click on any document in the list to display it.

A new window opens, displaying the document. If the document is too large to fit in the window, the section that shows the best match to the user question is displayed. One can display as many of the documents as one likes and can also save and print documents, using commands in the FILE menu. The TEXT menu contains commands that control the format in which documents are displayed.

3. Searching from keywords alone is rarely sufficient. At this point, therefore, the user probably wants to select some documents (or some sections in documents) to guide the next step of her search. On the other hand, the user may have found what she wanted; or, she may not have time at the moment to pursue the search. In these cases, she can either discard the question or save it to reuse at a later time.

Improving questions: The user can use the information from one set of results to modify her original question or to create further questions. Improving the original question often leads to improved results. So, if the first search did not provide the desired results, try one of the following:

- Probably the best way to improve a question is to drag one or more particularly interesting documents from the Results list into the “*Similar To*” field. When

the user runs the question again, the Results list will be updated with the new results, which will include documents similar to the one she selected (if such documents exist in the database).

- Alternatively, if one particular section within a document seems more useful than the rest, one can ask WAISStation to search for documents that resemble that section. To do that,
 1. Drag the cursor over a section of the document to select the text. An icon will appear in the left margin.
 2. Drag the new icon into the “*Similar To*” field.
 3. Run the query again by pressing SEARCH. The results will be updated with documents similar to the selected text.
- Add or remove keywords. This is often helpful if WAISStation did not interpret the initial words in the expected way.
- Add or remove sources. Add sources by dragging them into the source well; remove them by dragging them out of the source well.

Following tangents:

- One can change the topic completely by changing keywords.
- One can follow a new line of research that suggests itself, by dragging the relevant results (whole documents or sections) into the “*Similar To*” field.

The user may want to change her keywords to fit the new topic as well. Any of these strategies can be accomplished either by altering the original question, or by creating new questions that preserve the context of the older questions. Chaining questions in this manner allows the user to follow interesting new directions in a search, without losing any of the old data.

Chaining questions:

1. While one question window is open, select NEW QUESTION from the QUESTION menu. The new Question window will contain the same sources as the existing Question window has. The user can then alter this list of sources, or use it as is.

2. Drag documents or sections of documents from the existing Results into the "Similar To" field in the new Question window. Or, type keywords into the new window. Or do both.
3. RUN the new question.

Relevance Feedback: Providing relevance feedback to WAISStation, as explained in Chapter 5, allows users to narrow the results of queries until the desired information is found. But it also permits us to widen a search, or change it completely. While the user is looking for information on Subject A, if an interesting article on Subject B turns up, she can select that article as the model for the next search. Moreover, she can branch off onto a new search without losing track of the original search. In fact, one can continue both searches, and even start others; the range and movement of the searches are up to the user.

To close/save a question:

1. Click on the box in its upper left corner. If the question has not been saved previously, or if the user has made changes to the question, WAISStation asks if she wants to save the changes.
2. Click on YES to save the new version of the question. If the question has not been saved before, it will appear in the QUESTIONS window.

Click on NO to discard the modifications (or, if this is a new question, to discard the entire question.)

To save a question without closing it: Select SAVE or SAVE AS from the FILE menu.

To re-open a question: Double-click on its icon in the QUESTIONS window. (Or, open the question from the QUESTION menu.)

Updating sources: The system administrator should arrange to have some or all sources updated at regular intervals. To find out when a source is updated, and to tell WAISStation when to search that source, use the SOURCE window for each source your questions search.

1. Double-click on a Source name (or choose OPEN SOURCE from the Source menu).
2. The SOURCE window for the selected source appears. In the upper half of the window it contains information about when the database for this source was last updated.
3. The user may select the number of document headlines she wishes to have displayed with the Number of Documents pop-up menu.
4. The user can also select the default font and size of the text for documents that are displayed from this source, using the Font and Size pop-up menus.
5. Click on the Contact field in the lower half of the window. The user can then choose how often she wants her questions to search this source. The selected interval will appear in the Contact field, followed by fields that request further information.
6. Click on those fields, and enter the information.

References

- [1] Thinking Machine Corporation made the WAIS product freely available from <ftp://quake.think.com/pub/wais>.
- [2] All information about Gopher may be obtained from <ftp://boombox.micro.umn.edu/pub/gopher> or by sending email to gopher@boombox.micro.umn.edu.
- [3] All information about www may be obtained from <http://info.cern.ch/pub/www>.
- [4] D. Clark R. Braden and S. Shenker. Integrated Services in the Internet Architecture. Available by anonymous FTP. From <ftp://ftp.cs.ubc.ca/pub/archive/mirror/rfc/rfc1633.txt>.
- [5] C. Adie. Network Access to Multimedia Information. Available by anonymous FTP. From <ftp://ftp.cs.ubc.ca/pub/archive/mirror/rfc/rfc1614.txt>.
- [6] Douglas Comer. *Internetworking with TCP/IP*, volume 1. Prentice-Hall, second edition, 1991.
- [7] Available by anonymous FTP. From ftp://archive.cis.ohio-state.edu/pub/internic/isoc/charts/metrics-gifs/14_hosts_reachable.gif.
- [8] Available by anonymous FTP. From ftp://archive.cis.ohio-state.edu/pub/internic/isoc/charts/metrics-gifs/08_internets_growth.gif.
- [9] Available by anonymous FTP. From ftp://archive.cis.ohio-state.edu/pub/internic/isoc/charts/metrics-gifs/14_hosts_reachable.gif.
- [10] Available by anonymous FTP. From <ftp://archie.mcgill.ca/archie/doc/whatis.archie>.

- [11] S. Carl-Mitchell and J. S. Quaterman. *Practical Internetworking with TCP/IP and Unix*. Addison-Wesley, 1993.
- [12] URI Archive. Available by anonymous FTP. From <ftp://info.cern.ch/pub/ietf/uri-archive.Z>.
- [13] Tim Berners-Lee. WWW Names and Addresses, URIs, URLs. WWW Page. From <http://info.cern.ch/hypertext/WWW/Addressing/Addressing.html>.
- [14] Universal Resource Identifiers. WWW Page. From <http://info.cern.ch/hypertext/WWW/Addressing/URL/URIOverview.html>.
- [15] Tim Berners-Lee. Uniform Resource Locators. WWW Page. From <http://info.cern.ch/hypertext/WWW/Addressing/URL/Overview.html>.
- [16] J. Reynolds and J. Postel. Assigned Numbers. Available by anonymous FTP, Oct 1994. From <ftp://ftp.isi.edu/in-notes/iana/assignments>.
- [17] Chris Weider and Peter Deutsch. Uniform resource names. IETF Working Draft, May 1993. From <ftp://ftp.cc.mcgill.ca/pub/Network/uri/draft-ietf-uri-urn-00.txt>.
- [18] Extract from the file *ls-lr*. Available by anonymous FTP. From <ftp://ftp.cc.mcgill.ca/ls-LR.Z>, on July 6, 1994.
- [19] Alan Emtage and Peter Deutsch. *Archie: An Electronic Directory Service for the Internet*, Proc. Usenix Conf., 1992. Sunset Beach, Calif.
- [20] Client software for Archie. Available by anonymous FTP. From <ftp://ftp.sura.net/pub/archie/clients>.
- [21] Archie. Unix Programmer's Manual.
- [22] Archie client for the X interface. Available by anonymous FTP. From <ftp://gatekeeper.dec.com/.3/net/infosys/archie/clients>.
- [23] Ed Krol. *The whole Internet: user's guide & catalog*. O'Reilly & Associates Inc., Sebastopol, Calif., 1992.
- [24] Brewster Kahle and Art Medlar. An Information System for Corporate Users: Wide Area Information Servers. Version 3, TMC Technical Report, TMC199, April 1991. From <ftp://think.com://pub/wais/doc/wais-corp.txt>.

- [25] M. St. Pierre, J. Fullton *et. all* . WAIS over Z39.50-1988. Available by anonymous FTP, June 1994. From <ftp://ddn.nic.mil/rfc/rfc1625.txt>.
- [26] Search with WAIS for "gif" on the weather server quake.think.com.
Source name: weather.src, Maintainer: weather-server@quake.think.com.
- [27] G. Salton. Another Look at Automatic Text-Retrieval Systems. *Communications ACM*, pages 648-656, July 1986.
- [28] Ken Arnold. Screen Updating and Cursor Movement Optimization: A Library Package. Unix Manual.
- [29] Available by anonymous FTP. From <ftp://wais.com/pub/wais-doc/how.to.swais.txt>.
- [30] WAIS free client software. Available by anonymous FTP. From <ftp://midway.uchicago.edu/pub/network-clients>.
- [31] Brewster Kahle. Overview of wide area information servers. Available by anonymous FTP, August 1991. From <ftp://quake.think.com/wais/doc/overview.txt>.
- [32] Ron Hall. Using Gopher. *McGill University Computing Centre Newsletter*, page 6, March/April 1993.
- [33] Veronica source. Available by anonymous FTP. From <ftp://ftp.cc.mcgill.ca/pub/network-services/gopher/Veronica.tar.Z>. More information and help requests can be addressed to gophadm@futique.scs.unr.edu.
- [34] Steven Foster cited by J. Armour Polly. Available by anonymous FTP. From <ftp://nysernet.org/pub/resources/guides/surfing.2.0.3.txt>.
- [35] Steven Foster and Fred Barrie. Frequently Asked Questions. Available by anonymous FTP, June 1993. From <ftp://ftp.cac.psu.edu/pub/doc/Veronica.FAQ>.
- [36] Free Gopher clients and servers. Available by anonymous FTP. From <ftp://boombox.micro.umn.edu/pub/gopher/NeXT>.
- [37] J.F. Groff T. Berners-Lee, R. Cailliau and B. Pollermann. World-Wide Web: The Information Universe. Available by anonymous FTP. From ftp://info.cern.ch/pub/www/doc/Article_9202.ps.Z.

- [38] WWW client for NeXT. Available by anonymous FTP. From <ftp://xcf.berkeley.edu/pub/ht/projects/WWW/WWWBook/WWW/NeXT/Implementation>.
- [39] *Internet Resource Discovery Services*, University of Southern California, 1993. From ftp://info.cern.ch/pub/www/doc/Katia_Kobraczk_paper.ps.Z.
- [40] Regional Information: Countries. WWW Page. From <http://akebono.stanford.edu/yahoo/RegionalInformation/Countries/>.
- [41] Paul Robinson. Host tables. Internet News, October 1993. alt.internet.services, message ID <299nmo\$dph@news1.digex.net>.
- [42] Gongyu Dai. MapView: a viewer for picture databases, August 1992. McGill University, School of Computer Science.
- [43] Available by anonymous FTP. From <ftp://nic.funet.fi/pub/netinfo/UUCP/uumap>.
- [44] Vincent Cate. Alex - a Global Filesystem. Available by anonymous FTP. From <ftp://alex.sp.cs.cmu.edu/doc/intro.ps>.