Internet Resource Discovery Services

Peter B. Danzig, Katia Obraczka and Shih-Hao Li
Computer Science Department
University of Southern California
Los Angeles, California 90089-0781
danzig@usc.edu 213-740-4780

Introduction

The exponentially growing global Internet is a virtual infinite fountain of information, yet it can seem like an information labyrinth [22] when you try to find any specific fact.

A few years ago a researcher starting a new research project probably would have paid for a bibliographic search, contacted his fellow researchers, and leafed through conference proceedings and university technical report lists. Today, the Internet has the potential of being the researcher's main information source, however, the Internet's size and complexity is a considerable obstacle.

Recently, a number of tools have been developed to assist users in finding information of interest. These *resource discovery* tools specialize in browsing, searching, and organizing information distributed throughout the Internet. Browsing tools allow users to navigate through the available information space, and find information of interest during this navigation process. Query-based search tools automatically locate relevant data for the user based on information about the user's interest. Independent of the approach used, discovery services can also provide users with the ability to organize information once found, so that in the future they can refer to it without having to repeat the entire discovery process.

This paper presents an overview of resource discovery services currently available on the Internet. First, we survey a number of existing Internet discovery services. Then, we present a taxonomy of design decisions and characteristics of tools for the Internet resource discovery problem [30].

The Wide Area Information Server

WAIS, the Wide Area Information Server project [13, 12] resulted from the combined effort of four companies with complementary interests in the information discovery and retrieval problem. One of the project's primary goals is to provide users with a uniform, easy-to-use, location-transparent mechanism to access information.

WAIS is a full text information retrieval architecture whose clients and servers communicate through an extension of the Z39-50 protocol standard [14, 25]. One server is distinguished as the directory of services. Figure 1 shows a schematic view of the WAIS architecture.

The WAIS client translates user queries into the WAIS protocol, and can query the WAIS directory of servers for relevant servers. The request is then transmitted to an appropriate set of servers. WAIS servers keep complete inverted indices on the contents of documents they store and execute full-text searches on them. In response to a query, a WAIS server returns a list of relevant object descriptors. These descriptors correspond to documents that contain words and phrases specified in the user query. The WAIS client displays the results of the query to the user and retrieves the selected documents from the corresponding servers. WAIS clients support relevance feedback to help users refine their queries.

Originally, users were supposed to keep track of the available WAIS servers by storing their addresses, a description

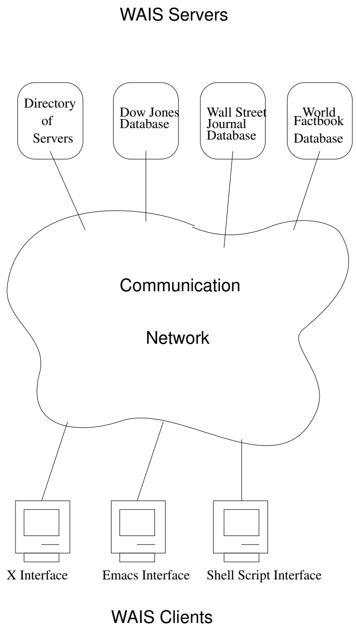


Figure 1: The WAIS Architecture.

of their contents, and if applicable, their accessing costs. However, as participating servers proliferate, it became unreasonable to expect that every user would keep track of all the available servers. A directory of servers service was made available. This *yellow page* service resides in a well-known address and is queried the same way as the regular WAIS servers. Instead of reference to documents, it replies with references to participating WAIS servers. When a new information provider wants to join WAIS, it must submit its location, description, and other relevant information to the directory server. Currently, there are 285 servers registered with the WAIS directory of services. The WAIS directory of servers's database is replicated in other servers and has its primary copy stored on host *think.com*.

An example WAIS discovery session using the X WAIS client is presented in Figure 2. The query is first submitted to the directory of services which responds with descriptors of pertinent servers. The user can then select servers from this set and submit the query to them. In this example, two servers are selected and queried. They return 40 different document descriptors. By looking at their descriptions, the user can identify relevant documents to be retrieved. If the user wishes to refine the search, one or more of the returned documents can be selected and placed in the `Similar to:` box. When the query is re-run, the results are updated to include documents similar to the selected ones. Similarity is ranked in terms of number of common words.

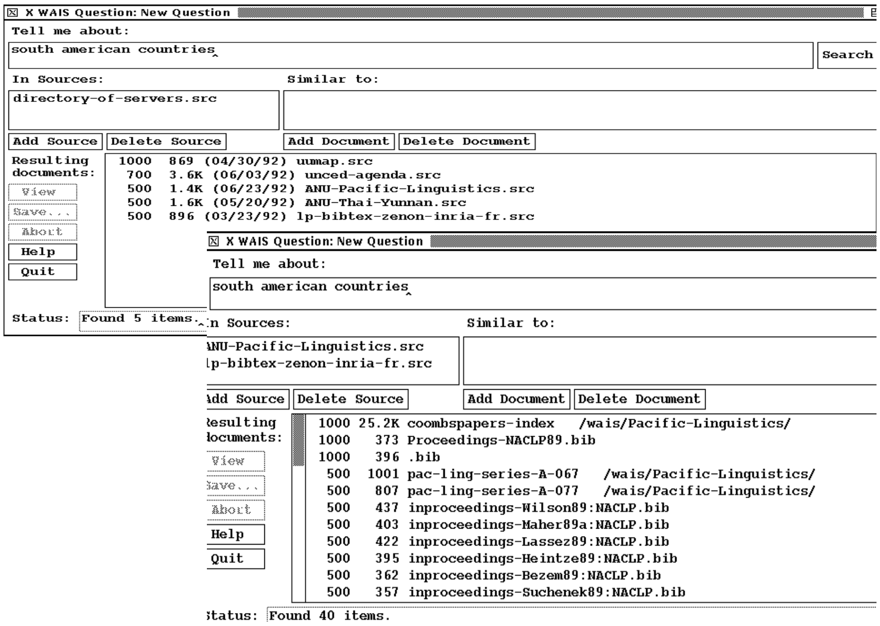
Archie

Archie [9] specifically addresses the file discovery and retrieval problem in large internetworks. Archie servers centralize indexing information on file name data distributed throughout Internet archive sites.

The current archie service makes available two databases. One of them, the *filenames* database indexes names of files available from hundreds of FTP sites [1]. Those entries are automatically updated by archie. Users can query

---

[1] FTP is the Internet File Transfer Protocol, which is used to retrieve files stored on Internet hosts. Anonymous FTP is is a special FTP session, which does not require the user to have an account on the FTP site. Anonymous FTP is a widely used method to support distribution of public domain software and documents.



Tell me about: south american countries

In Sources: directory-of-servers.src

Similar to:

Resulting documents:

```
 1000   869 (04/30/92) uumap.src
  700  3.6K (06/03/92) unced-agenda.src
  500  1.4K (06/23/92) ANU-Pacific-Linguistics.src
  500  1.6K (05/20/92) ANU-Thai-Yunnan.src
  500   896 (03/23/92) lp-bibtex-zenon-inria-fr.src
```

Status: Found 5 items.

Tell me about: south american countries

In Sources: ANU-Pacific-Linguistics.src
lp-bibtex-zenon-inria-fr.src

Similar to:

Resulting documents:

```
1000 25.2K coombspapers-index   /wais/Pacific-Linguistics/
1000   373 Proceedings-NACLP89.bib
1000   396 .bib
 500  1001 pac-ling-series-A-067   /wais/Pacific-Linguistics/
 500   807 pac-ling-series-A-077   /wais/Pacific-Linguistics/
 500   437 inproceedings-Wilson89:NACLP.bib
 500   403 inproceedings-Maher89a:NACLP.bib
 500   422 inproceedings-Lassez89:NACLP.bib
 500   395 inproceedings-Heintze89:NACLP.bib
 500   362 inproceedings-Bezem89:NACLP.bib
 500   357 inproceedings-Suchenek89:NACLP.bib
```

Status: Found 40 items.

Figure 2: An Example WAIS Discovery Session.

this database for file names that match specified patterns, a complete list of the FTP archive sites, or a list of the files available from specific sites.

The other database, the *whatis* database, contains the names and descriptions of software packages, documents, and other information available on the Internet. Entries in the *whatis* database are text strings, consisting of keywords and the associated descriptions. Users perform case-insensitive text string searches that are applied to both the keywords and the corresponding descriptions. Currently, the *whatis* database is manually maintained. That is, information is gathered from secondary sources, such as Usenet postings, and author's e-mail submission, and entered into the database by hand.

The archie architecture is presented in Figure 3. The data gathering (DGC) and maintenance (DMC) components maintain the *filenames* database, while the *whatis* database is maintained manually. Archie clients access both databases through the user access component (UAC). The data gathering component relies on FTP site administrators to find out about new FTP archives. Every time a new FTP archive is reported, an entry corresponding to the the new site is added to the file that lists all known FTP sites, the *site descriptions file*. Periodically, the DGC connects to each known FTP site, and fetches a recursive listing of its available contents. This information is kept on the archie server in the *raw listing files*, and is processed by the DMC, which converts the raw site listings into a format that can be added to the *filenames* database.

The user access component allows users on the Internet to access and query the archie servers. Currently, there are three possible access methods: the *telnet* interface, the *e-mail* interface, and the *Prospero* [18, 17, 16, 19] interface. The telnet interface to archie consists of connecting to an archie server through the *telnet* command, and then submitting queries. Users can then submit queries to either the *whatis* and *filenames* databases. The telnet interface has proven to be very resource consuming, since each *telnet* session requires a significant amount of server resources. As an alternative, the Prospero interface has been recently made available to users. It consists of having a Prospero server as a front-end to the archie server. The Prospero server allows Prospero users to access archie's databases without the need to log onto the archie server directly. The Prospero interface has also caused the development of archie clients using the Prospero protocol. Because of Prospero's UDP based protocol and the Prospero server scheduling and caching techniques, the Prospero interface to archie is less resource consuming and presents higher

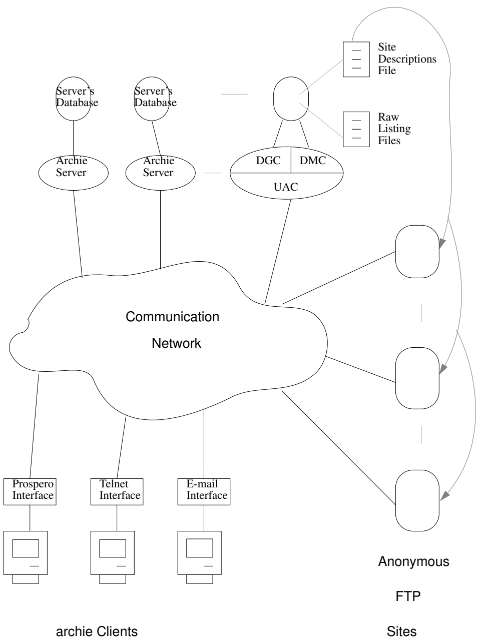


Figure 3: The Archie Architecture.

throughput than the telnet access. Through the e-mail interface, users can submit a query to archie by sending an e-mail message to an archie server, which sends a message back to the user with the query results.

There are currently nine archie servers on the Internet, five of which provide general access to the whole Internet community, and the other four serve communities with limited Internet access. Consistency among the archie servers is maintained by having them copy the site listing information kept at the main archie server in Montreal, Canada. According to [9], the traffic generated by the update mechanism and user queries to the main archie server is responsible for approximately 50% of all Montreal-bound Internet traffic. This could be avoided by distributing the work of polling the participating sites among the various archie servers, and then have them execute a consistency maintenance protocol among themselves.

Archie's developers have described it as a low-tech solution to the resource discovery and information retrieval problem. However, archie's simplicity and use of existing mechanisms have been the key to its success. After a little more than a year in service, with over 1,000 registered FTP archive sites, making available 2,100,000 files and 3,500 different software packages, archie is accessed from 47 different countries, reporting 3,500 search queries per day at the Canada server.

Prospero

The Prospero File System [18, 17, 16, 19] is a tool for organizing distributed information. Prospero lets users build customized views, or *virtual systems*, of directories distributed throughout the Internet.

The Prospero name space forms a generalized directed graph, where intermediate nodes are directories, leaves are files, and edges are Prospero links. Just like traditional distributed file systems, subtrees of the Prospero name space can be stored by different Prospero servers. A user's name space corresponds to the subgraph starting at a particular node, which is the root of the user's name space.

Users organize their name space hierarchically by building views, which are essentially directories composed from various sources including the user's own views and imported views from other users. These directories may reside in different Prospero servers. One special type of view is an index, which returns a directory of objects that satisfy some query. This allows Prospero users to access other search engines. For example, the Prospero-archie interface lets users build views containing directories with objects resulting from archie queries.

Users can find information by navigating through available views. The Prospero client provides users with navigational tools that are analogous to the ones provided by traditional file systems. One command allows a user to change his current virtual directory to the one specified in the command. Another command displays the name of the current virtual directory and describes its physical location. Users can also list the contents of a virtual directory.

A sample session using Prospero is shown in Figure 4. This particular Prospero session illustrates the Prospero-archie interface. The user starts by using the `vfsetup` command which places the user at a specific point in the Prospero name space, in this case, the `guest` virtual system. Then through `vcd`, `vwd`, and `vls` the user can change his current virtual directory, display the name of the current virtual directory, and list its contents, respectively. To submit an archie request to find all file names that match the string *wais*, the user changes directory to */databases/archie/regex/wais*, and lists the results by using the `vls` command.

When a user finds an interesting object, which may be a simple file or a view, the user can include the object in his view by linking to it. The Prospero client provides commands to add and delete links from the current node in the user's name space to a target node or leaf. A Prospero link specifies the name of the host where the object is stored and the local name of the object on that host. If the target of the link is a directory, the link provides information to resolve a name in that directory by querying the corresponding server. For files, the associated links contact the appropriate server to provide access mode information. Currently, Prospero supports Sun's Network File System, the Andrew File System, and anonymous FTP. A link also includes information such as link type, and filters associated with the link. Special links allow the contents of the target directory to be virtually included in the physical directory containing the link. By associating filters with links, users can build customized views from existing ones. A filter customizes the target view by reorganizing or extracting parts of it.

In summary, listing a Prospero view requires a computation distributed across all of the nodes reachable by transitive closure of all of the view's links, indices, and filters.

Users advertise information by registering their virtual system with the Prospero server administrator. The administrator creates a link to the new virtual system in the master view of virtual systems, where other users can see, navigate, and if accessible, link to portions of it.

Currently, there are close to 50 Prospero servers. Most users from more than 10,000 systems in 30 different countries run as Prospero clients only. They either run the Prospero interface to archie or the full Prospero client.

Gopher

The Internet Gopher [1] allows users to search and browse distributed information. Gopher organizes information into a hierarchy [2], where intermediate nodes are directories or indices, and leaf nodes are documents.

The Gopher architecture is composed of clients and servers communicating through the Gopher protocol, which is implemented on top of TCP-IP. Figure 5 shows the Gopher architecture.

The root of Gopher's hierarchy is stored on host *rawBits.micro.umn.edu* at the University of Minnesota. This is the default directory retrieved by the Gopher client when first invoked. Gopher clients can also be configured with other entry points into the Gopher hierarchy. The Gopher root server knows about all top-level services, so that it can advertise their existence to users. In the Gopher architecture, there is essentially one *top-level server* per participating organization, such as a university campus, a private corporate institution, or a government agency. *Lower-level servers* may be linked to the corresponding top-level server, so that once users find the appropriate top-level server, they can navigate through the hierarchy by following the links to the lower-level servers. For example, university campuses running Gopher servers may register a central top-level server with the Gopher root server. Each university's central Gopher server may have links to existing departmental servers, which in turn may have links to

---

[2] Actually, the Gopher information space is a generalized directed graph, since it allows cycles.

```
% vfsetup guest
% vwd
/guest

% vcd ..
% vls
VIRTUAL-SYSTEMS PROSPERO.ISI.EDU /pfs/pfsdat/guest/local_vsystems/guest
VS-DESCRIPTION PROSPERO.ISI.EDU /pfs/pfsdat/guest/local_vsystems/guest
afs PROSPERO.ISI.EDU /afs
databases PROSPERO.ISI.EDU /pfs/pfsdat/guest/local_vsystems/guest
documents PROSPERO.ISI.EDU /pfs/pfsdat/guest/local_vsystems/guest
guest PROSPERO.ISI.EDU /pfs/pfsdat/guest/local_vsystems/guest
info PROSPERO.ISI.EDU /pfs/pfsdat/guest/local_vsystems/guest
lib PROSPERO.ISI.EDU /pfs/pfsdat/guest/local_vsystems/guest
mailing-lists PROSPERO.ISI.EDU /pfs/pfsdat/guest/local_vsystems/guest
newsgroups PROSPERO.ISI.EDU /pfs/pfsdat/guest/local_vsystems/guest
papers PROSPERO.ISI.EDU /pfs/pfsdat/guest/local_vsystems/guest
projects PROSPERO.ISI.EDU /pfs/pfsdat/guest/local_vsystems/guest
releases PROSPERO.ISI.EDU /pfs/pfsdat/guest/local_vsystems/guest
sites PROSPERO.ISI.EDU /pfs/pfsdat/guest/local_vsystems/guest

% vcd databases
% vls
archie PROSPERO.ISI.EDU /pfs/pfsdat/prototype_g/databases/arch
bibliographies PROSPERO.ISI.EDU /pfs/pfsdat/prototype_g/databases/bibl
faces NIC.SURA.NET ARCHIE/HOST/ftp.uu.net/faces
machine-learning JUNE.CS.WASHING /u1/vfs/pfsdat/local_vsystems/pablo/pa

% vcd archie
% vls
au PROSPERO.ISI.EDU /pfs/pfsdat/prototype_g/databases/arch
mcgill PROSPERO.ISI.EDU /pfs/pfsdat/prototype_g/databases/arch
regex ARCHIE.MCGILL.CA ARCHIE/MATCH
sura PROSPERO.ISI.EDU /pfs/pfsdat/prototype_g/databases/arch

% vcd regex
% vcd wais
% vls
wais ARCHIE.MCGILL.CA ARCHIE/HOST/ftp.uu.net/pub/archiving/
wais-8-b3-ses.tar.Z ugle.unit.no /pub/gopher/unix/wais-8-b3-ses.tar.Z
wais-8-b3.README burgundy.nectar /readmes/readmes.untared/gr/forth/csi/
wais-8-b3.tar.Z fly.bio.indiana /util/wais/wais-8-b3.tar.Z
wais-minutes-92mar.t aramis.rutgers. /internet-drafts/.ietf/wais-min-
utes-92
wais.library ariel.unm.edu /library/wais.library
wais.src bitsy.mit.edu /pub/alpha-beta-dos/potluck/wais/wais.
wais.txt biox.bioz.uniba /archive_data/database/info/various/wa
wwais102.zip bitsy.mit.edu /pub/alpha-beta-dos/potluck/wais/wwais
```

Figure 4: A Sample Prospero Session (Some of the server and path names are truncated).

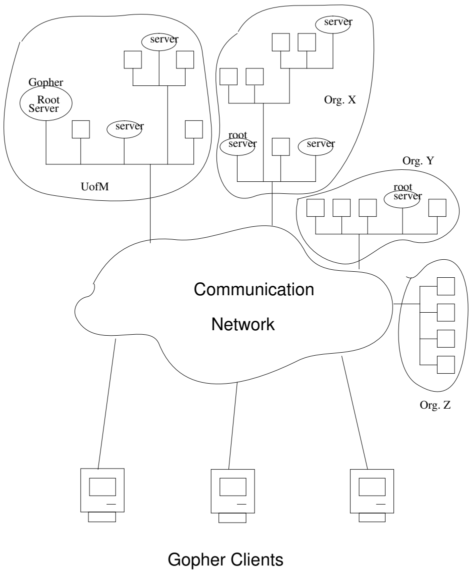


Figure 5: The Internet Gopher Architecture.

lower-level servers.

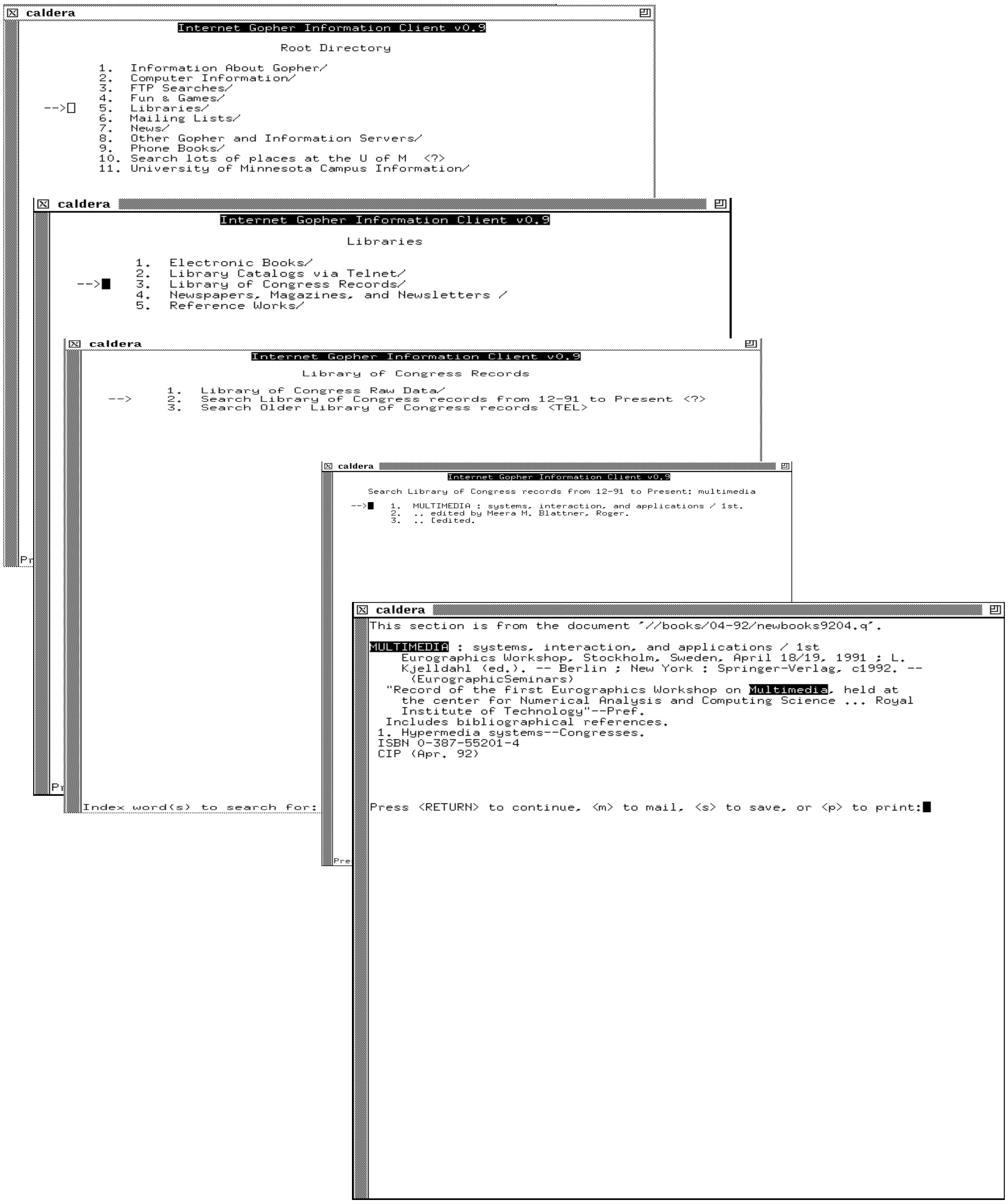
Gopher objects are identified by their type, user-visible name, server's host name and port number, and the object's absolute path name within the server's file system. The user selects an object based on its user-visible name, and the Gopher client retrieves it by constructing a handle from the server's host name and port number, and the object's path name. Users can then navigate through the available information base containing full-text document objects, which are stored as files in the corresponding servers, and directory objects that may be distributed across multiple servers. Full-text search operations can also be performed. Gopher's *search servers* maintain full-text inverted indices of subsets of the documents stored in a Gopher server. Search servers can be configured to index more than one server. For instance, there is an Internet Request for Comments (RFC) full-text search server that indexes all the existing RFC's, and executes keyword searches on their contents. A full-text search server returns to the client handles to documents that match a Boolean search pattern. Gopher clients can also retrieve objects from WAIS, archie, and FTP servers.

Figure 6 illustrates a sample Gopher session. Starting at the root directory, the user traverses the Gopher information space by selecting interesting directories, such as `Libraries/` and `Library of Congress Records/`, or executing full-text searches on indices like `Search  Library of Congress Records from 12-91  to Present <?>`.

The World-Wide Web

The World-Wide Web, or WWW [3, 2] merges the techniques of information discovery and hypertext. WWW organizes data into a distributed hypertext, where nodes are either full-text objects, directory objects called *cover pages*, or indices. WWW also supports full-text searches over documents stored at a particular WWW server.

The WWW architecture is also based on the client-server model. The WWW client provides users with a hypertext-like browsing interface. Besides its native HyperText Transfer Protocol, WWW clients understand FTP



```
                 Internet Gopher Information Client v0.9

                              Root Directory

          1.  Information About Gopher/
          2.  Computer Information/
          3.  FTP Searches/
          4.  Fun & Games/
   -->    5.  Libraries/
          6.  Mailing Lists/
          7.  News/
          8.  Other Gopher and Information Servers/
          9.  Phone Books/
          10. Search lots of places at the U of M  <?>
          11. University of Minnesota Campus Information/
```

```
                 Internet Gopher Information Client v0.9

                                Libraries

          1.  Electronic Books/
          2.  Library Catalogs via Telnet/
   -->    3.  Library of Congress Records/
          4.  Newspapers, Magazines, and Newsletters /
          5.  Reference Works/
```

```
                 Internet Gopher Information Client v0.9

                       Library of Congress Records

          1.  Library of Congress Raw Data/
   -->    2.  Search Library of Congress records from 12-91 to Present <?>
          3.  Search Older Library of Congress records <TEL>

Index word(s) to search for:
```

```
                 Internet Gopher Information Client v0.9

    Search Library of Congress records from 12-91 to Present: multimedia

   -->    1.  MULTIMEDIA : systems, interaction, and applications / 1st.
          2.  .. edited by Meera M. Blattner, Roger.
          3.  .. [edited.
```

```
This section is from the document ´//books/04-92/newbooks9204.q´.

MULTIMEDIA : systems, interaction, and applications / 1st
    Eurographics Workshop, Stockholm, Sweden, April 18/19, 1991 ; L.
    Kjelldahl (ed.). -- Berlin ; New York : Springer-Verlag, c1992. --
      (EurographicSeminars)
  "Record of the first Eurographics Workshop on Multimedia, held at
    the center for Numerical Analysis and Computing Science ... Royal
    Institute of Technology"--Pref.
  Includes bibliographical references.
 1. Hypermedia systems--Congresses.
 ISBN 0-387-55201-4
 CIP (Apr. 92)



Press <RETURN> to continue, <m> to mail, <s> to save, or <p> to print:
```

Figure 6: A Sample Gopher Session.

and the Network News Transfer Protocol, NNTP [11]. FTP is used for accessing file archives on the Internet, where file directories are browsed as hypertext objects. NNTP allows access to Internet news groups and news articles. News articles may contain references to other articles or news groups, which are represented as hypertext links.

HTTP allows document retrieval and full-text search operations. HTTP runs on top of TCP and maps each request to a TCP connection. HTTP objects are identified by the HTTP protocol type, the corresponding server's name, and the path name to the file where the objects' contents reside. Parts of documents can also be specified. If a search operation is requested, the HTTP object identifier carries the set of specified keywords, instead of a path name. Future implementations of HTTP protocol will include data format negotiation between client and server. Currently, only plain text and simple hypertext formats (HyperText Markup Language- HTML) are implemented.

Information accessible through WWW can be seen through three discovery trees. One tree is a classification by subject. An entry in the WWW root directory links to the current subject classification tree. Currently this classification includes topics such as aeronautics, astronomy, biological sciences, computer sciences, and humanities. This information is spread across all kinds of servers, including WAIS, Gopher, and WWW. Because WWW was created for the high-energy physics community, a special entry in the root directory links to a cover page with the existing HTTP servers specialized in the subject. As they become available, indices to other disciplines will be added. The other WWW discovery tree is a classification by server type. The cover page corresponding to this classification lists all the servers available through WWW. This includes entries for WAIS, Gopher, NNTP and WWW servers. There is even an entry for anonymous FTP sites that are searched through archie. The third tree is a classification by organization and is not very populated.

The WWW discovery trees correspond to the different ways information is organized and can be discovered. Discovery sessions involve users starting on their home cover page, following a link to an index, then executing a search, and following the resulting links. As users find interesting information, they build their personalized web by linking to nodes in the global web. Currently, the default cover page, which resides on host *info.cern.ch* and represents the root of the WWW information space, is the one retrieved by the WWW client when first invoked. However, users can customize their home cover page so that they can start anywhere in the WWW information space. Figure 7 presents an example of a WWW session.

Making information available through WWW may involve a similar discovery task. The information publisher must try to find the appropriate cover page that should reference the new data. Then the publisher should contact the person responsible for that cover page so that a link to the new data is added. Another possibility is for the publisher to run a new server. This last option requires that the new server's administrator contacts the WWW administrators to add the new server to the list of existing servers.

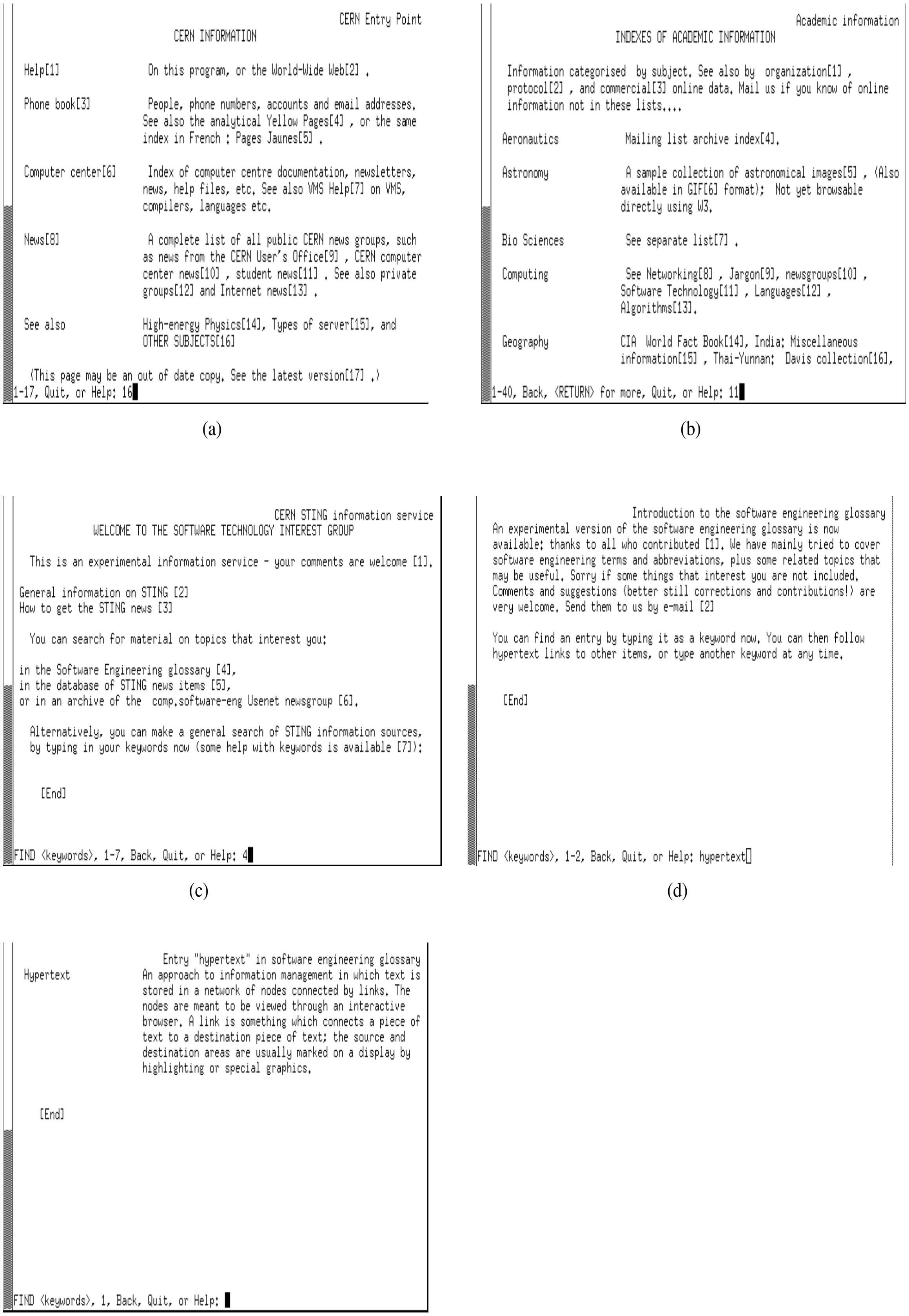
In summary, WWW is a hypertext-like tool for organizing and accessing information available Internet-wide. Using its browsing interface, users can navigate through the different classification trees to find and publish new information.

Currently, besides WAIS and Gopher servers, there are about 24 WWW servers accessible to WWW clients. The WWW server on *info.cern.ch* has logged access from approximately 6,000 different hosts that use their own WWW clients or connect to the publically available client.

Resource Discovery at the University of Colorado

The Resource Discovery project at the University of Colorado-Boulder [29, 27] is best known for *netfind*, but has also investigated various issues in the resource discovery arena. Netfind [26] is a white pages directory tool which given an Internet user's name and organization, tries to locate available information about the user. A successful netfind query, like the one shown in Figure 8 returns information such as the user's e-mail address, and telephone number.

Netfind builds its indexing database, called the *seed database*, based on data scattered across multiple existing sources, such as network news messages, the Domain Name System (DNS), the Simple Mail Transfer Protocol, and the *finger* utility. The seed database keeps organization names, city names, and corresponding host names gathered from news message headers over time. Based on the organization names and city names provided to netfind, matching host names are selected from the seed database. DNS is then used to locate authoritative name servers for the domains to which the selected hosts belong. Each of the name servers found are queried using SMTP to try to find



```
                                                    CERN Entry Point
                              CERN INFORMATION

 Help[1]                 On this program, or the World-Wide Web[2] .

 Phone book[3]           People, phone numbers, accounts and email addresses.
                        See also the analytical Yellow Pages[4] , or the same
                        index in French : Pages Jaunes[5] .

 Computer center[6]      Index of computer centre documentation, newsletters,
                        news, help files, etc. See also VMS Help[7] on VMS,
                        compilers, languages etc.

 News[8]                 A complete list of all public CERN news groups, such
                        as news from the CERN User's Office[9] , CERN computer
                        center news[10] , student news[11] . See also private
                        groups[12] and Internet news[13] .

 See also               High-energy Physics[14], Types of server[15], and
                        OTHER SUBJECTS[16]

  (This page may be an out of date copy. See the latest version[17] .)
1-17, Quit, or Help: 16█
```

(a)

```
                                                          Academic information
                         INDEXES OF ACADEMIC INFORMATION

  Information categorised  by subject. See also by  organization[1] ,
  protocol[2] , and commercial[3] online data. Mail us if you know of online
  information not in these lists....

 Aeronautics             Mailing list archive index[4].

 Astronomy               A sample collection of astronomical images[5] , (Also
                        available in GIF[6] format);  Not yet browsable
                        directly using W3.

 Bio Sciences            See separate list[7] .

 Computing               See Networking[8] , Jargon[9], newsgroups[10] ,
                        Software Technology[11] , Languages[12] ,
                        Algorithms[13].

 Geography              CIA  World Fact Book[14], India: Miscellaneous
                        information[15] , Thai-Yunnan:  Davis collection[16].

1-40, Back, <RETURN> for more, Quit, or Help: 11█
```

(b)

```
                                                CERN STING information service
         WELCOME TO THE SOFTWARE TECHNOLOGY INTEREST GROUP

  This is an experimental information service - your comments are welcome [1].

General information on STING [2]
How to get the STING news [3]

  You can search for material on topics that interest you:

in the Software Engineering glossary [4],
in the database of STING news items [5],
or in an archive of the  comp.software-eng Usenet newsgroup [6].

  Alternatively, you can make a general search of STING information sources,
  by typing in your keywords now (some help with keywords is available [7]):


    [End]


FIND <keywords>, 1-7, Back, Quit, or Help: 4█
```

(c)

```
                            Introduction to the software engineering glossary
An experimental version of the software engineering glossary is now
available: thanks to all who contributed [1]. We have mainly tried to cover
software engineering terms and abbreviations, plus some related topics that
may be useful. Sorry if some things that interest you are not included.
Comments and suggestions (better still corrections and contributions!) are
very welcome. Send them to us by e-mail [2]

You can find an entry by typing it as a keyword now. You can then follow
hypertext links to other items, or type another keyword at any time.


  [End]


FIND <keywords>, 1-2, Back, Quit, or Help: hypertext▯
```

(d)

```
                           Entry "hypertext" in software engineering glossary
 Hypertext              An approach to information management in which text is
                        stored in a network of nodes connected by links. The
                        nodes are meant to be viewed through an interactive
                        browser. A link is something which connects a piece of
                        text to a destination piece of text; the source and
                        destination areas are usually marked on a display by
                        highlighting or special graphics.


    [End]


FIND <keywords>, 1, Back, Quit, or Help: █
```

(e)

Figure 7: A Sample WWW Session.

mail forwarding information about the specified user. If found, the corresponding hosts are probed using *finger*. To improve netfind's response time and increase its resiliency to host and network failures, up to ten lightweight threads may be used to allow sets of DNS, SMTP, and finger query sequences to be executed in parallel.

Other resource discovery related projects under development at the University of Colorado include a probabilistic yellow page tool [28], a network visualization tool [29], which focuses on discovering information about networks, such as topology, congestion, routing, and protocol usage, and a global electronic mail study [31], which investigates the organization of human social networks by analyzing mail logs collected from different Internet sites.

Indie

Distributed Indexing, or Indie for short [5, 6], is a distributed information discovery and retrieval architecture. Indie consists of a replicated *directory of services* and a collection of *broker* databases that automatically cluster references to related information by indexing their own data, and data stored in other brokers, databases, and other discovery tools. This clustering of indexing information in Indie brokers makes efficient exhaustive search possible. Furthermore, because it was built atop of the Distributed Hypertext (DHT) system data model and communication protocol [20], Indie inherits the organizational capabilities of hypertext systems. Thus users can also benefit from Indie's ability to organize information of interest, so that it can be easily found next time the user needs it.

An Indie broker stores descriptors of objects that are relevant to the topic the broker specializes in. These object descriptors are extracted from other Indie brokers and primary data sources that the broker indexes. An *object descriptor* contains an arbitrary number of attribute-value pairs describing the object. Examples of attributes that constitute an object descriptor include bibliographic information about the object, such as the author's name, the document's title and publication date, an abstract and keywords describing the document's contents. The object descriptor also includes technical data about the object, such as the object type, the object identifier and timestamp assigned to each object by the database that created it, and some attributes used by Indie's consistency maintenance mechanism. The object descriptors need not include the object itself. This lets a service advertise an object but retain control of access to it.

A *generator object* describes an Indie broker. The generator consists of a textual abstract, a boolean expression over a set of bibliographic fields, which we call the *generator rule*, and fields such as the broker's location, the size of its database, and the object identifier corresponding to the generator object. The name *generator rule* expresses the idea that a broker's database is generated and periodically updated by evaluating the rule over a number of other brokers and primary data sources.

To become visible to users and other brokers, all brokers register their generator object with Indie's directory of services. A broker can register itself with any replica of the directory of services. As part of the registration procedure, the selected replica returns a list of other generator objects pertinent to the new broker. The broker stores this list in its *registration table*, and refers to it when choosing the set of databases it will index. The directory of services replica also reports changes to the broker's registration table, as new brokers join in or participating brokers cease to exist. The broker administrator, via the administrator's tool, selects brokers to index from its registration table. From time to time, the administrator refers to the registration table and decides whether or not to index new brokers.

Brokers that are indexed by others store the indexing brokers' generator objects in their *trigger table*. The name *trigger table* is an analogy with active databases [33] in which specific rules are triggered and evaluated when the database changes in particular ways. When one broker registers its generator object with another broker, the indexed broker executes the generator rule and reliably forwards the retrieved set of object descriptors to the indexing broker. Afterwards, adding or deleting objects from the indexed broker's database may trigger the evaluation of generator rules in its trigger table, and some of these rules may forward these changes to the corresponding indexing brokers.

The directory of services is just a specialized broker. When a broker registers itself with a replica of the directory of services, the broker's generator object is stored in that replica's trigger table. Only the updates to the directory of services that trigger this rule are forwarded to the broker.

Non-Indie servers attach to Indie through a *gateway* broker. Indie provides a *gateway library* consisting of a set of routines which non-Indie servers can call to communicate with their gateway broker. The gateway broker itself

```
caldera.usc.edu% telnet bruno.cs.colorado.edu
Trying 128.138.243.151 ...
Connected to bruno.cs.colorado.edu.
Escape character is `^]'.

SunOS UNIX (bruno)

login: netfind

=====================================================
Welcome to the University of Colorado Netfind server.
=====================================================

I think that your terminal can display 24 lines.
If this is wrong, please enter the "Other" menu and
set the correct number of lines.

Help/Search/Other/Quit [h/s/o/q]: s
Enter person and keys (blank to exit) --> goldweic chicago

There are too many domains in the list.
Please select at most 3 of the following:
 0. 308.mct.anl.gov (argonne national laboratory, chicago, illinois)
 1. a136.mct.anl.gov (argonne national laboratory, chicago, illinois)
 2. acc.uic.edu (academic computing center, university of illinois, chicago)
 3. adv.com (advantage design group, chicago, illinois)
 4. amoco.com (amoco corporation, chicago, ill)
 5. andersen.com (andersen consulting, chicago, illinois)
.
.
.
 88. uchicago.edu (university of chicago, illinois)
.
.
.

Enter selection (e.g., 3 1 2) --> 88
( 0) check_name: checking domain uchicago.edu. Level = 0
( 0) get_domain_addr: Got nameserver oddjob.uchicago.edu
( 0) get_domain_addr: Got nameserver gargoyle.uchicago.edu
( 0) check_name: checking nameserver oddjob.uchicago.edu. Level = 2
( 1) check_name: checking nameserver gargoyle.uchicago.edu. Level = 2
MAIL FOR Patricia Goldweic IS FORWARDED TO goldweic@gargoyle.uchicago.edu
SYSTEM: gargoyle.uchicago.edu
 Login name: goldweic In real life: Patricia Goldweic
 Directory: /homes/goldweic Shell: /bin/tcsh
 Last login Thu May 28 15:41 on ttype from hope.uchicago.ed
 New mail received Mon Jul 13 15:41:11 1992;
 unread since Fri Jul 10 17:15:40 1992
 Plan:
 Office : 702-8584 #256
 Home : 728-3827

 Address:
 4600 N. Clarendon Av. #605
 Chicago, IL. 60640-5748

Continue the search ([n]/y) ? -->
Enter person and keys (blank to exit) -->
Help/Search/Other/Quit [h/s/o/q]: q
Exiting Netfind server...
```

Figure 8: A Sample Netfind Session.

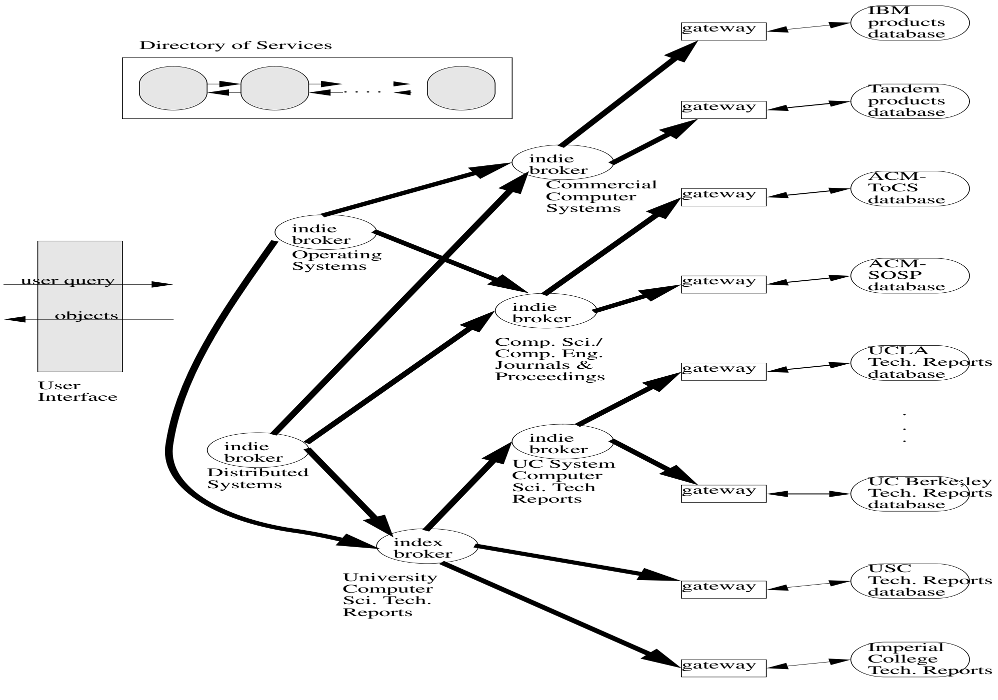


Figure 9: Indie Operation.

is just a normal Indie broker modified as necessary to communicate with non-Indie servers. It manages a trigger table, registration table, and the interface to the directory of servers. A non-Indie server can efficiently attach itself to Indie by making appropriate calls to the gateway library. Otherwise, the gateway broker must communicate with the non-Indie server in its native protocol, and poll for updates by periodically extracting indexing information from the server.

Figure 9 illustrates an example indexing configuration and its operation. Suppose that a user wants to find all recent technical publications on Distributed Operating Systems. The corresponding user query is submitted to a replica of the directory of services, which evaluates the query against its generator object database. The result of this computation is a list of brokers whose descriptions are pertinent to the user query's focus. In this example, the directory of services could return a reference to the Operating Systems broker, and the Distributed Systems broker. The user interface ranks the list of target brokers according to their interest. This ranking procedure could be based for instance on counting the number of keyword matches in the broker descriptor. In this case, the user interface could send the query to both brokers, whom after evaluating the user query on their indexing information database, return a list of appropriate object descriptors. Based on this information, the user may choose to retrieve a copy of one or more interesting articles from one or more full text retrieval systems. For instance, the user decides to retrieve a copy of an interesting report from UCLA. The user interface contacts the corresponding gateway to retrieve the selected object. Alternatively, the user could have chosen to add a link from any node in her information space to the UCLA technical report, so that she could find it and refer to it in the future without having to repeat the entire discovery process.

Indie addresses database consistency and recovery with a timestamped augmented flooding algorithm which also permits convenient recovery from network partition, operating system crashes, and media failure of the broker's database. Indie's timestamp based consistency mechanism works as follows. All trigger table entries, and registration table entries are timestamped. When an object is added to or deleted from a broker, the change causes the broker to evaluate certain rules stored in its trigger table against updates with timestamps younger than the trigger table entry. The broker forwards the appropriate changes to each of its affected *peers*, that is, each registered indexing broker corresponding to the triggered generator rules, and advances the trigger table timestamps to the timestamp of the latest update. It then transmits this timestamp to each of the affected peers, which record it in the appropriate registration table entry. When the indexed broker cannot establish communication with one peer, it marks the trigger table entry as out of date. Timestamps of rules neither triggered nor marked as out of date are advanced to the current time. Finally, peers occasionally poll one another in an attempt to maintain consistency. The timestamp mechanism permits convenient recovery from network partition, operating system crashes, and media failure of the broker's database.

Lazily consistent broker replication is a side effect of Indie's indexing mechanism. To replicate a broker, we create a new broker to serve as the replica, assign the replica the same generator rule as the broker to be replicated, and have the replica index the broker or some number of replicas of it. Since the replica shares the same generator rule as the primary copy, it fills with the same data. Indie's update and recovery algorithm guarantees that all replicas eventually learn of the update.

Replication in the context of the directory of services considers all replicas to be equal. This means that there is no primary copy of the directory of servers. Instead, clients register or unregister themselves with the replica of their choice. All replicas participate in a flooding based consistency maintenance mechanism to keep their databases consistent.

Indie's first implementation phase has been completed. Currently, the prototype consisting of indie brokers, a centralized directory of services and gateways to FTP archives is running on the USC Network and Distributed Systems Laboratory. Indie's consistency maintenance and replication mechanisms are now being implemented.

Other Resource Discovery Research Initiatives

Besides the services described above, other discovery tools include X.500, the Knowbot Information Service, Alex, Semantic File Systems, and Nomenclator.

X.500 [32] is the result of the CCITT (Consultative Committee for Telephony and Telegraphy) and ISO (International Standards Organization) standardization efforts in the field of directory of services. X.500's name space is hierarchically organized and distributed among X.500 servers. Administrative authority over portions of the global name space is delegated to different autonomous organizations, who can in turn transfer authority over portions of their assigned subtrees. Unlike the Domain Name System [15], X.500 accepts attribute-based queries. The original X.500 standard specification does not explicitly mention support for replication. However, QUIPU [24], one of X.500's current implementations, uses a simple replication mechanism based on designated slave and master servers.

The Digital Library System, or DLS [10] is an open architecture whose goal is to integrate access to all existing as well as future information sources available on the Internet. Knowbots, the abbreviation for Knowledge Robots, are active components of the DLS. A Knowbot is defined as an active intelligent program capable of exchanging messages with other Knowbots, moving and replicating itself around the system, searching and manipulating objects. Based on the Knowbot concept, the Knowbot Information System (KIS) [8] understands a number of directory services, such as X.500, and query these services on behalf of users.

Alex [4] is a file system that provides users with transparent read access to files in Internet anonymous FTP sites. Through Alex, users see the collection of Internet anonymous FTP sites and their corresponding directories and files as a hierarchical file system, where intermediate nodes are Internet domains, hosts, or directories within hosts, and leaves are files. Using the standard filesystem commands, users can browse through this hierarchy and retrieve files of interest. To get reasonable performance, Alex caches information such as machine names, directory information, and the contents of remote files. Alex implements a soft consistency mechanism which guarantees that only updates that occurred in the last 5% of the reported age of the file might not yet been reflected locally. Alex is currently implemented as an NFS server, and already integrates access to archie. WAIS servers that index README files and computer science technical reports available through Alex have also been built.

The Semantic File System [7] integrates associative access into a traditional tree-structured file system. Associative access is achieved by providing file systems with an attribute extraction and query interface. Attribute extraction is performed by filters called *transducers*. A transducer takes as input the contents of a file or of a directory, and produces as output the identifiable objects and their corresponding attributes. An object may correspond to an entire directory, a file, or portions of a file, such as procedures in a source code file, or individual messages in a mail file. Queries consist of boolean combinations of the desired attribute-value pairs. Transducers and queries produce

Service	Query	Browse	Organize	Granularity	Information Space		Directory of Services
					Organization	Distribution	
Prospero		•	•	Files	Generalized Dir. Graph	Distributed	
Gopher		•		Files	Generalized Dir. Graph	Distributed	
WWW		•	•	Documents	Generalized Dir. Graph	Distributed	
Semantic File System	•	•	•	Portions of files	Generalized Dir. Graph		
X.500	•		•	Info about users		Distributed	
Alex		•	•	Files	Generalized Dir. Graph	Distributed	
Archie	•			File names	Indices	Replicated	
WAIS	•			Documents	Indices	Distributed	•
Netfind	•			Info about users	Indices	Replicated	
Indie	•		•	Documents	Indices	Distributed	•
Nomenclator	•			Same as X.500	Indices	Distributed	

Table 1: A Taxonomy for Internet Resource Discovery Services.

customized views of the file system hierarchy called *virtual directories*, which help locate and organize information. A semantic file system research prototype has been implemented on top of Sun NFS.

Nomenclator [21, 23] implements an attribute-based, or *yellow-page* naming on top of hierarchical naming systems. Nomenclator *access functions* are, in essence, servers that periodically traverse the appropriate portions of the underlying name space and other access functions. They build indices of the objects encountered that satisfy certain properties. The Nomenclator client uses a directory of services called the *active catalog* to identify access functions pertinent to a user's query. A Nomenclator prototype that uses X.500 as its underlying information repository has been built.

Conclusions

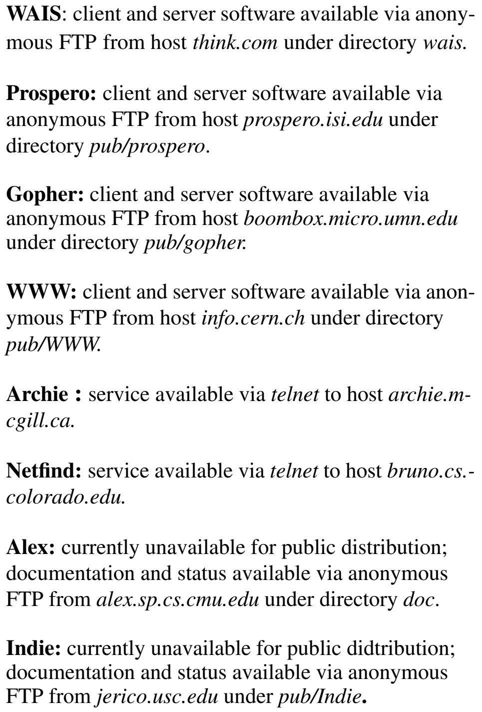
Motivated by the continually growing number of hosts on the Internet accompanied by the corresponding increase in the amount of available information, Internet resource discovery services have proliferated. This paper surveyed prototyped discovery and retrieval tools currently available on the Internet.

We summarize the surveyed tools by presenting a taxonomy of approaches to the resource discovery problem. Table 1 lists a number of features according to which we classify the surveyed discovery tools.

Services like Gopher and WWW provide users with a browsing interface with which they can navigate through the available information space. WWW is also an organizational tool. Like Prospero, Alex, and traditional file systems, WWW organizes its information space using links. By customizing their home cover pages, WWW users can link to interesting information anywhere in the WWW information space. However, users can only customize their starting point in the WWW space, having to follow existing links from then on. Because it is file-system oriented, Prospero is a more flexible organizational tool. It allows users to customize their entire information space using Prospero links and filters. Nevertheless, unlike WWW and hypertext systems in general, in which any node can be linked to any other node, Prospero can only link a directory node to another directory node or to a file node.

When responding to a user's query, services such as archie, WAIS, netfind, and Indie search their indexing databases for relevant information. These tools build their indexing databases from information distributed throughout the Internet. Because it is built atop the Distributed Hypertext (DHT) scheme, Indie also has the potential for allowing users to organize their information space into a distributed hypertext.

The granularity with which discovery tools manipulate objects is another distinguishing feature. In archie for example, target objects are file names, instead of file contents. Therefore, archie can only be used to locate information stored in files that have meaningful names. In the case of WAIS, which indexes the contents of documents, users can



WAIS: client and server software available via anonymous FTP from host *think.com* under directory *wais.*

Prospero: client and server software available via anonymous FTP from host *prospero.isi.edu* under directory *pub/prospero.*

Gopher: client and server software available via anonymous FTP from host *boombox.micro.umn.edu* under directory *pub/gopher.*

WWW: client and server software available via anonymous FTP from host *info.cern.ch* under directory *pub/WWW.*

Archie : service available via *telnet* to host *archie.mcgill.ca.*

Netfind: service available via *telnet* to host *bruno.cs.colorado.edu.*

Alex: currently unavailable for public distribution; documentation and status available via anonymous FTP from *alex.sp.cs.cmu.edu* under directory *doc.*

Indie: currently unavailable for public didtribution; documentation and status available via anonymous FTP from *jerico.usc.edu* under *pub/Indie***.**

Figure 10: How to access Internet resource discovery services or where to find their software packages.

find interesting documents by submitting keyword-based queries. In the Semantic File System approach, users can access self-contained portions of a file, such as procedures in a source code file, or individual messages in a mail file.

Discovery tools organize searchable data into some kind of information space. Usually, browsing tools organize their information space as a generalized directed graph with nodes connected by links. Prospero, Gopher, WWW, and Alex belong to this category. WWW is a step forward towards hypertext systems. It allows links between any kind of nodes. On the other hand, query-based search services tend to organize searchable information into indexing databases, which makes efficient exhaustive search possible.

Besides how discovery tools organize data, there is also the question of where this data is stored. In services that employ a graph-based organization, data is usually distributed among geographically dispersed servers. Prospero, Gopher, and WWW belong to this group of services. On the other side of the design spectrum, tools like Archie and Netfind build centralized and replicated indexing databases. Indie and WAIS build distributed indexing databases. WAIS servers store both the indexing database and the corresponding data. In Indie, indexing databases are distributed among Indie brokers according to the topic Indie brokers specialize in.

Both Indie and WAIS have implemented directory of services. Discovery sessions usually start with a query to the directory of services, which provides users with hints on places to search. The WAIS directory of services knows about all the participating WAIS servers, and when responding to a user's query, provides a list of relevant servers. Similarly, Indie's directory of services responds to users' queries with a list of relevant Indie brokers. For scalability and availability purposes, Indie's directory of services implements Indie's replication mechanism.

The initial development of existing Internet discovery tools was done independently from each other. However, the current trend points in the direction of interoperability, where users of one discovery service can access information available through other services. For instance, Gopher clients can retrieve objects from archie, WAIS, and FTP servers. WWW clients understand FTP archives, Gopher servers, and archie. Currently, the resource discovery interoperability efforts include the specification of uniform document identifiers. In the near future, we believe that users will benefit from an integrated Internet resource discovery fabric.

References

[1] R. Alberti, F. Anklesaria, P. Lindner, M. McCahill, and D. Torrey. The Internet Gopher protocol: a distributed document search and retrieval protocol. On-line documentation, Spring 1992.

[2] T. Berners-Lee, R. Cailliau, J-F. Groff, and B. Pollermann. World-Wide Web: An information infrastructure for high-energy physics. *Proceedings of the Workshop on Software Engineering, Artificial Intelligence and Expert Systems for High Energy and Nuclear Physics*, January 1992.

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

[4] Vincent Cate. Alex-a global filesystem. On-line documentation, April 1992.

[5] Peter Danzig, Jongsuk Ahn, John Noll, and Katia Obraczka. Distributed indexing: A scalable mechanism for distributed information retrieval. *Proceedings of the 14th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 220–229, October 1991.

[6] Peter B. Danzig, Shih-Hao Li, and Katia Obraczka. Distributed indexing of autonomous internet services. *Submited to Journal of Computer Systems*, June, 1992. Available via anonymous FTP from jerico.usc.edu:pub/Indie/jcs.ps.Z.

[7] Mark A. Sheldon David K. Gifford, Pierre Jouvelot and James W. O'Toole Jr. Semantic file systems. *Proceedings of the 13th ACM Symposium on Operating Systems Principles*, pages 16–25, October 1991.

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

[9] Alan Emtage and Peter Deutsch. archie: An electronic directory service for the internet. *Proceedings of the Winter 1992 Usenix Conference*, January 1992.

[10] Robert E. Kahn and Vinton G. Cerf. The digital library project Volume 1: The world of Knowbots. Technical report, Corporation for national Research Initiatives, 1988.

[11] B. Kantor and P. Lapsley. Network news transfer protocol - a proposed standard for the stream-based transmission of news. Internet Request for Comments RFC 977, February 1986.

[12] Brewster Khale. Wide area information server concepts, alpha release documentation. Available via FTP from think.com:wais/wais-8-b5.tar.Z, April 1991.

[13] Brewster Khale and Art Medlar. An information system for corporate users: Wide area information servers. *ConneXions - The Interoperability Report*, 5(11), November 1991.

[14] Clifford A. Lynch. The Z39-50 information retrieval protocol: An overview and status report. *ACM Computer Communication Review*, 21(1):58–70, 1991.

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

[16] B. Clifford Neuman. The Prospero file system user's manual. Technical Report, Department of Computer Science and Enngineering, University of Washington, Seattle, Washington, January 1991.

[17] B. Clifford Neuman. The Prospero file system: A global file system based on the virtual system model. Technical Report, Department of Computer Science and Enngineering, University of Washington, Seattle, Washington, July 1991.

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

[19] B. Clifford Neuman. Prospero: A tool for organizing internet resources. *Electronic Networking: Research, Applications and Policy*, 2(1), Spring 1992.

[20] John Noll and Walt Scacchi. Integrating diverse information space repositories: A distributed hypertext approach. *IEEE Computer*, 24(12):38–45, December 1992.

[21] Joan J. Ordille and Barton P. Miller. Active cataloguing and caching in a descriptive name service. Technical Report, Computer Science Department, University of Wisconsin-Madison, Madison, Wisconsin, 1991.

[22] Joan J. Ordille and Barton P. Miller. Lost in a labyrinth of workstations. Workshop on Workstations Operating Systems, June 1992.

[23] Joan J. Ordille and Barton P. Miller. Nomenclator descriptive query optimization for large X.500 environments. *ACM SIGCOMM 91 Conference*, pages 185–196, September 1991.

[24] C. Robbins and S. Kille. *The ISO Development Environment: User's Manual, Volume 5: QUIPU*, September 13, 1991.

[25] Martin L. Schoffstall and Wengyik Yeong. A critique of Z39-50 based on implementation experience. *Computer Communication Review*, 20(2):22–29, April 1990.

[26] Michael F. Schwartz. Experience with a semantically cognizant internet white pages directory tool. *Journal of Internetworking Research and Experience*, 1(2), December 1990.

[27] Michael F. Schwartz. Resource discovery and related research at the university of colorado. Technical Report CU-CS-508-91, Department of Computer Science, University of Colorado, Boulder, Colorado, January 1991.

[28] Michael F. Schwartz. A scalable, non-hierarchical resource discovery mechanism based on probabilistic protocols. Technical Report CU-CS-474-90, Department of Computer Science, University of Colorado, Boulder, Colorado, June 1990.

[29] Michael F. Schwartz. Resource discovery in the global internet. Technical Report CU-CS-555-91, Department of Computer Science, University of Colorado, Boulder, Colorado, November 1991.

[30] Michael F. Schwartz, Alan Emtage, Brewster Kahle, and Clifford Neuman. A comparison of internet resource discovery approaches. Technical Report CU-CS-601-92, Department of Computer Science, University of Colorado, Boulder, Colorado, July 1992.

[31] Michael F. Schwartz and David C. M. Wood. A measurement study of organizational properties in the global electronic mail community. Technical Report CU-CS-482-90, Department of Computer Science, University of Colorado, Boulder, Colorado, August 1990.

[32] B. Smetaniuk. Distributed operation of the X.500 directory. *Computer Networks and ISDN Systems*, pages 17–40, 1991.

[33] Michael Stonebraker and Lawrence A. Rowe et al. The postgres papers. Technical report, UC Berkeley, June 25, 1987.