# Content Routing in a Network of WAIS Servers

Andrzej Duda*
Mark A. Sheldon

Programming Systems Research Group
MIT Laboratory for Computer Science

## Abstract

Locating and accessing information in a large distributed system is a difficult problem of growing importance. This paper reports on our experience building and using a prototype system for transparent, user-guided associative access to the contents of a large, distributed set of WAIS servers. Our system is based on *content routing*, an architecture that makes use of *content labels* for locating and accessing information in large distributed systems [12]. Our content router for WAIS servers is implemented as a Semantic File System that constructs content labels from WAIS source and catalog files. The content router guides locating documents by suggesting terms that frequently appear with a given query term in document headlines. Sufficiently narrowed queries are routed to WAIS servers and processed in parallel. We have successfully used our content router to locate documents on a large number of WAIS servers. Along with demonstrating the feasibility of distributed finding in a large network of information servers, the prototype shows interactive performance limited primarily by the processing time at WAIS servers.

## 1 Introduction

Locating and accessing information in a large distributed system is a difficult problem of growing importance. The size of the Internet, along with the development of high-speed networks, has engendered an increasing number of accessible information servers. These servers represent a vast potential resource. However, the lack of associative access to this collection makes it difficult to discover relevant information.

The Wide Area Information Service (WAIS) [8, 14] is one of the more widely used information systems on the Internet. In October 1993, there were 492 WAIS servers covering such domains as Usenet news, computer science technical reports, biology citations, and zip codes. WAIS was designed for searching one or several servers using relevance feedback: the user starts with some *seed* words, browses through documents containing these words, and optionally specifies that she wants to find other documents similar to a given document. This interactive approach can be fairly efficient when the user knows which server to query. However, it is not well suited for querying a large number of servers of unknown content and for browsing and exploring the global information space. The primary means for discovering relevant WAIS servers is to query a special server called the *directory of servers*. It holds a succinct description of each server, called a *source file*. As the description in a source file only indicates a general statement of the information domain covered by a given server, it is difficult to find which servers are relevant to a query. For example, the search for the term `video` in the directory of servers gives a list of 6 relevant servers, whereas there are many more: a brute force search revealed over 270 servers that contained documents with the term `video`.

Another problem with querying a large number of servers with the WAIS software is that searching is done sequentially. The increasing elapsed time of a search on multiple servers makes user interaction more difficult.

In spite of these problems, the aggregate of all WAIS servers forms a large, interesting information space. To fully exploit this space, we need a means for discovering relevant servers, for routing queries to them, and for guiding query formulation. The need for guiding query formulation results from the vastness of the information space — existing WAIS servers cover so many distinct, heterogeneous domains that finding something useful is difficult.

Our approach to locating and accessing information

This work was supported by the Defense Advanced Research Projects Agency and the Department of the Army under contract DABT63-92-C-0012.

* Also with Bull-IMAG Systèmes and INRIA

in a large distributed system is based on *content routing*, which we have described in a previous paper [12]. A *content routing system* (CRS) defines a distributed architecture for content-based navigation and associative access to documents. A content routing system is organized as a network of servers called *content routers* that present a single query based image of a distributed information system. Content routers support a *collection* document type that may be thought of as an extension to a traditional information retrieval system. A collection is a set of documents together with a description of the set called a *content label.* Content labels serve for routing queries to relevant servers and for discovering the global information space. From the user's perspective, a content router appears to be an information server that contains all the information in its collections. In addition to query routing, a content router provides query refinement that helps a user formulate sufficiently narrow queries. If the user wants to discover and explore the information space, she starts with a broad search term and the system suggests possible completions to that term. When a completion is chosen, the server presents to the user the content labels of relevant collections. When a sufficiently narrow query is formulated, it is routed to relevant collections.

This paper reports our experience building and using a prototype content router for WAIS servers. Our content router for WAIS servers is implemented as a Semantic File System [6] that can be mounted by any NFS client. For our initial tests, we chose to construct content labels from WAIS source and catalog files. WAIS source files are very brief descriptions of information sources, often listed at a special directory of servers. A source file contains host name, host address, database name, port number, and often a short list of keywords and/or a human readable description of the database's purpose. WAIS catalog files are lists of headlines, one for each document in the database. Thus, they contain information-rich terms that characterize document content well.

The content router guides locating documents of interest by suggesting terms that frequently appear with a given query term in the document headlines. Sufficiently narrowed queries are sent to WAIS servers and processed in parallel. Results are merged and presented to the user. Initial experience with our prototype suggests that a content routing system is an effective tool for exploring and searching a network of WAIS servers.

In the remainder of this paper we review related work (Section 2), remind the content routing architecture (Section 3), present an implementation of a content router for WAIS servers (Section 4), describe some practical experience (Section 5), and outline conclusions based on our experience (Section 6).

## 2 Related work

Previous work can be broken down into two broad categories: network navigation systems and network-based information retrieval systems.

Network navigation systems such as the Gopher system [1] and the Word-Wide Web [3] provide a means for organizing servers into a structure that allows navigating among remote servers and browsing their contents. If the user knows what to look for and where the resource might be located, then interesting resources can be discovered. These systems provide facilities for individual servers that respond to queries. There is no support for query routing.

Network-based information retrieval systems provide associative access to information on remote servers. We have already introduced WAIS. The Archie system [5] polls a fixed set of FTP sites on the Internet. A query yields a set of host/filename pairs which is then used to manually retrieve the relevant files. The Conit system [9, 10] provides a uniform user interface to a large number of databases accessible from several retrieval systems. User queries are translated into commands on the different systems. However, there is no support for the automatic selection of relevant databases for a user's query. The Distributed Indexing mechanism [4] is based on precomputed indices of databases that summarize the holdings on particular topics of other databases. The architecture has a three layer fixed structure that is suitable for bibliographic databases, but it is not flexible enough for content based access to general information servers. Simpson and Alonso propose a querying architecture similar to ours for a network of autonomous databases [13] that forwards user queries to known information sources. The knowledge about existing sources is acquired by exchanging messages with neighboring hosts. The Information Brokers of the MITL Gold project [2] share many of our goals. However, Information Brokers do not provide access to objects directly. Rather, they return descriptions of an object's location and the method that may be used to retrieve the object. Nomenclator [11] is an architecture for providing efficient descriptive (attribute-based) naming in a large Internet environment. Active catalogs constrain the search space by providing information about the data distribution. Both responses to queries and data distribution information are cached to speed future queries.

The GLOSS system [7] uses a probabilistic scheme for forwarding queries to information servers. It explores a statistical approach to characterizing the contents of an information server by extracting a histogram of its words occurrences. The histograms serve for estimating the result size of each query that is used to choose appropriate information servers for searching.

The architecture of a content routing system and a brief description of the first prototype is given in our previous paper [12]. The content routing architecture combines efficient associative access to distributed information servers with query refinement. Key advances offered by the system include scalability, usability and efficiency.

The present work implements the content routing architecture in a large network of WAIS servers. This companion paper deals with the design, construction, and experience with the content router for WAIS servers. The implementation is the first attempt known to the authors of building a transparent, user-guided distributed information system on top of WAIS servers.

## 3 Content Routing Architecture

To make the paper self-contained, we provide below an overview of the main concepts of the content routing architecture. For a more complete description of the data model and semantics, see [12].

### System organization

A content routing system may be thought of as an extension to a traditional information retrieval system. It provides associative access to a set of documents. However, in addition to more traditional document types such as text, electronic mail, and video footage, a CRS supports a document type called a *collection*. A collection is a set of documents (which can also be collections) together with a description of the set called a *content label*. A content label is a query predicate that abstracts the contents of member documents. The documents of a collection satisfy the content label of that collection. A user of a CRS views a large collection as a hierarchical arrangement of documents in a directed acyclic graph (see Figure 1).

Architecturally, a CRS is composed of a network of information servers. Leaf nodes in the network are end-point information servers that support the base (conventional) document types. The hierarchical network of internal nodes is composed of *content routers*, which are information servers that support

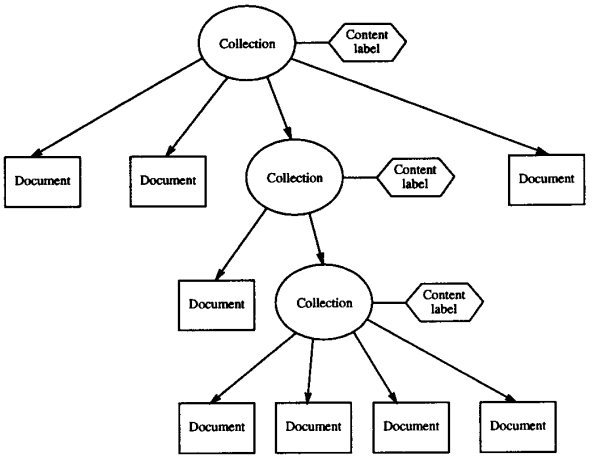


Figure 1: A content routing system provides access to a hierarchy of collections.

collection documents. This network mirrors the hierarchical organization of Figure 1. Information servers register with content routers by providing an interface for query and document retrieval requests as well as a content label.

A CRS provides operations for *expanding* a set of collections into its component documents, for *searching* collections, and for query completion (*show fields*, *show values*). A user may *select* a set of documents from a collection using a query, *list results* of a selection, and *retrieve* a document. Query or retrieval requests can be applied to content labels or routed to servers implementing collections.

### Query and content label semantics

A *query* is a boolean combination of *attributes*. Each attribute consists of a *field* name and a *value*. Examples of attributes are `text:retrieval` and `author:kahle`. When a query is applied to a set of documents, the subset that satisfies the query is called the *query result set*. In the process of query refinement, the user can enumerate the set of defined fields and browse the set of possible values for any field.

A content label is a query predicate that abstracts the contents of a collection. For example, a very small content label might be:

```
(collection-type:software) and ((domain:cmu.edu)
or (domain:stanford.edu)) and (subject:operating-systems)
```

This content label implies that every document at the server has attribute `collection-type:software`

and **subject:operating-systems** and is either in the **cmu.edu** or the **stanford.edu** domains.

The CRS design does not limit the form of content labels. In principle, a content label could consist of the disjunction of the attributes of all collection member documents. Alternatively, a content label could simply be the name of the host containing a set of documents. The first approach implies large content labels, but allows finding all documents relevant to a given query. The second approach results in small content labels, but is more suited for finding whole collections (for example, the documents in the collection can only be found using the query "find all documents on a given host"). In practice, content labels will lie between these two extremes, containing attributes including host names, domains, authors, libraries, subjects, and text. They will usually include two kinds of attributes: administratively determined *value added* attributes that describe a collection but may not appear in the documents themselves, and attributes derived from the collection contents, possibly using various statistical tools.

### System operation

Starting at the top level, a user may explore the logical organization of information servers. Using content labels, the system generates a view of the search space implied by a query. The user iteratively refines the query until enough discriminating power is present. A content router provides query refinement that helps a user formulate sufficiently narrow queries by suggesting possible completions to a query. When a completion is chosen, the server presents to the user the content labels of relevant collections. When the search space is sufficiently limited, the user may expand a set of collection documents into its component documents. The iterative process continues with queries being forwarded to the remote servers implementing the selected collections. Finally, the user may retrieve documents of interest.

## 4 Content Routing for WAIS Servers

This section presents a prototype content router for WAIS servers. The goal of building a content router for WAIS servers was to exploit the large information space created by emerging WAIS servers and to explore how content labels can be generated automatically for large collections of documents. We wanted to validate the content routing architecture using a large scale, real life example and gain experience on how content labels can be helpful for query formulation and for discovering resources in a large information space. Moreover, we wanted to verify if all these goals can be achieved with adequate performance.

Content routing based on content labels lies between the extremes of a global index and sending every query to every server. At the one extreme, one could obtain all index files of WAIS servers, construct a global index and use it for routing. Such a global index might guarantee perfect recall of relevant servers — all servers having a document relevant to a query can be found. However, the space needed for this approach is prohibitively large, and a global index is very difficult to keep up to date. In addition, there would still be some need to organize the heterogeneous data for browsing. At the other extreme, queries can be routed to all WAIS servers. However, the perfect recall of relevant servers is obtained at the expense of high communication and processing overhead. In addition, browsing is again difficult, both because it is time consuming and because the very high recall is potentially overwhelming. Obviously, neither solution scales well.

Our prototype incorporates several simplifications. Because our experiment involved using nearly 500 WAIS servers over which we have no control, it was not possible to enlist the aid of WAIS administrators to produce value added attributes. Nor was it possible to do detailed statistical analysis to chose the content-based attributes because we did not have access to the WAIS index structures (and browsing all the documents was not feasible). In a real content routing system, the individual servers would be responsible for this analysis. We therefore chose to experiment with very simple content labels that consisted of disjunctions of attributes obtained from the individual WAIS source and catalog files.

Our prototype uses the Semantic File System (SFS) path name interface described in [6]. The SFS pathname interface currently supports only conjunctive queries, though the underlying SFS query processing software does support arbitrary boolean queries. For the presentation here, we have used a straightforward transcription of our sessions into a more readable syntax. All the figures and data in this paper are derived from experiments performed during October 1993 when there were 492 WAIS servers available.

The remainder of this section describes the gateway between the SFS system and WAIS, and presents the content router for WAIS.

### 4.1 SFS-WAIS Gateway

Since WAIS servers do not use the SFS interface, we implemented an SFS-WAIS gateway. The SFS-WAIS gateway translates path name queries into WAIS ques-

tions and uses the public domain WAIS client code for querying WAIS servers. Conjunctive queries are of special importance because they narrow the search space. However, since many WAIS servers do not implement conjunction, our gateway implements the boolean **AND** operator by running a WAIS query for each query term and calculating the intersection of the result sets. Although it is computationally expensive (the server must find several, possibly large, result sets), it was the only way to use all existing WAIS servers. The result documents are viewed as files. (Actually, they appear as symbolic links. When a document file is accessed, the **readlink** NFS call is intercepted and the document is retrieved from the WAIS server.) A document name is constructed from the headline of the document obtained as a result of the query. Here is an example of querying the **zipcodes** WAIS server via the gateway:

```
=> Select  source:zipcodes and text:warsaw
=> List-results
14569_Warsaw__NY_.1     22572_Warsaw__VA_.2
28398_Warsaw__NC_.3     41095_Warsaw__KY_.4
43844_Warsaw__OH_.5     46580_Warsaw__IN_.6
55087_Warsaw__MN_.7     62379_Warsaw__IL_.8
65355_Warsaw__MO_.9
```

## 4.2 A Content Router for WAIS

As shown in the following examples, queries at the content router apply to the content labels for the servers registered there.

```
=> Select  text:buddhism
=> List-results
ANU-Asian-Religions.cl          Tantric-News.cl
ANU-Shamanism-Studies.cl        file-archive-uunet.cl
ANU-SocSci-Netlore.cl           lists.cl
ANU-Thai-Yunnan.cl              mailing-lists.cl
ANU-ZenBuddhism-Calendar.cl
ANU-ZenBuddhism-Listserv.cl
Omni-Cultural-Academic-Resource.cl
```

The content router uses the query to determine the set of collections. When the user *expands* the set of collections, the server forwards the query to the corresponding set of servers supporting the collections and presents the merged results to the user. This is useful for browsing the graph of information servers and/or refining a query so that fewer servers need to be contacted. To provide access to individual documents and servers, the content router returns symbolic links that are interpreted by the automount daemon at the client. The client then uses its own mount point that directly connects to the SFS-WAIS server.

While building a query, a user can learn about possible terms by using the query completion feature. She can list the set of known fields and the values of any field available in documents that match the specified portion of the query.

### 4.2.1 Content Labels

We decided to construct content labels from WAIS source and catalog files. The WAIS catalog files contain headlines for each document (the subject of a message or the title of an article). Thus, they contain information-rich terms that characterize document content well. Here is a sample from a catalog (from the Roget's thesaurus WAIS server):

```
Catalog for database: ../wais-documents/roget-thesaurus
Date: Aug 21 13:39:06 1992
1025 total documents
Document # 1
Headline: This is the 12/91 Project Gutenberg
release of Roget's Thesaurus
DocID: 0 8062 /u3/wais/roget/out/roget10.out
Document # 2
Headline:       #1. Existence.-- N. existence, being,
entity, ens, esse, subsistence
DocID: 8062 9508 /u3/wais/roget/out/roget10.out
Document # 3
Headline:       #2. Inexistence.-- N. inexistence;
nonexistence, nonsubsistence;
DocID: 9508 10859 /u3/wais/roget/out/roget10.out
```

The space requirement of a catalog file is only a fraction of the space of an entire WAIS database. Most WAIS servers return a catalog file in response to an empty query so that it can be retrieved automatically. As opposed to catalog files, WAIS source files contain only a minimal description of a server (the median size was under 800 bytes). They can serve to categorize a server into a general domain such as biology, economy or computer science, but they can hardly be used for discovering servers relevant to a query.

From 492 existing WAIS servers, we have retrieved 371 catalog files which occupy 176.4MB. The content of catalog files varies enormously: some headlines are very informative while others contain file path names or other less useful information. The largest catalog file takes 10.7MB and contains 71336 headlines; the smallest one is 79 bytes without any headlines. As only part of the catalog files (headlines) is used for constructing content labels and only unique terms are kept, the total size of the content labels is reduced to 6.1MB. Figure 2 presents the size distribution of catalogs and content labels.

Here is the start of a sample content label constructed from a WAIS source file and a catalog file:

```
hostaddress: 130.95.128.1
hostname: uniwa.uwa.oz.au
library: netinfo-docs
```

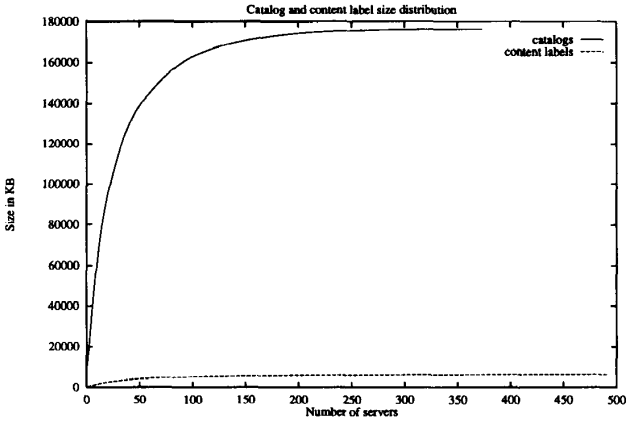


Figure 2: Catalog and content label size distribution

| term | source files | content labels | exhaustive search |
|---|---|---|---|
| carcinoma | 0 | 5 | 34 |
| discovery | 6 | 23 | 223 |
| multimedia | 9 | 45 | 189 |
| video | 6 | 62 | 273 |

Table 1: Number of relevant servers

```
label: Server created with WAIS release 8 b4 on
Mar 16 17:04:07 1992 by root@uniwa
Various files with information about
accessing the internet and what services
are available.
text: aarnet aarnetuser acadlist agriculture alias
anzlibraries archie architecture bailey bib
binaryftp bowers business cache canadian
card changes cmc compress consumer contacts
coombs coombspapers crossroads cruise cwis
description dir direct doc docindex draft
...
```

To evaluate how good are the content labels constructed from WAIS source and catalog files, we collected some statistics on the results of search using different information about the contents of WAIS servers. Table 1 shows how many servers were found using source files, our prototype's content labels, and a brute force search of all servers looking for documents containing the specified term. The server recall capability of content routing depends on the definition of a relevant server. If a relevant server is defined by terms in document headlines or in WAIS source files, then our prototype server has 100% recall by definition. If server recall is defined by terms in document text, then in Table 1, server recall varies between 10% and 22%. This recall rate could be substantially improved with enhanced WAIS server support for content label generation.

Even with the limitations of our present content labels, we think that content routing finds servers that are more relevant than others. This is because the terms obtained from headlines are more important than terms found in a document body. In a subsequent experimental trial using the term `carcinoma`, our system found 19% of the relevant servers using content labels. However, these servers provided 40% of all relevant documents. It is certainly the case that a better statistical analysis of a collection would yield better content labels and better recall.

### 4.2.2 Query Completion

An important feature of a content router is *query completion*, which guides query formulation. When a query term has been specified, this feature provides hints for possible terms that can be used to narrow the search space.

As in any SFS, the user may list the available fields and enumerate the values for a given field:

```
=> Show-fields
administrator:    immutable-field:  owner:
label:            port:             field:
server-field:     hostaddress:      library:
hostname:         location:     cost:
library-type:     text:
=> Show-values  library:
aarnet          eurogopher        paris
aas             factbook          pathology
aboriginal      factsheets        pegasus
abs             falcon            performance
abstracts       faq               philippine
academic        faqs              phone
academics       fax               phones
acronyms        fits              pictures
address         fixes             plant
addresses       flight            poetry
...
```

The guidance provided to the user through this interface is very important for browsing a large set of information servers. To implement this in our prototype, we had the content router build a WAIS index of all headlines from the catalog files so that the headlines containing a given term can be found efficiently. The size of the database is 362MB. We are currently investigating a variety of techniques to represent query completion data more compactly and integrate them with content labels.

When a user asks for a list of fields or for the values of a field, the server uses the query completion database to find all the headlines containing the current query terms. It finds the set of most frequent

terms collocated with the query terms. These new terms are presented to the user.

Query completion can be used in formulating a query. In the example below, the content router suggests terms related to the user's query term **text:buddhism**.

```
=> Select  text:buddhism
=> Show-values  text:
academic  chinese   data  religion   society
asian     cultural  omni  religions  tibetan
bases     culture   raw   resource   zen
```

If the user chooses a term, relevant WAIS servers can be discovered by inspecting content labels:

```
=> Select  text:buddhism and text:tibetan
=> List-results
ANU-Asian-Religions.cl      ANU-SocSci-Netlore.cl
ANU-Shamanism-Studies.cl    Tantric-News.cl
file-archive-uunet.cl
```

When the user decides that the search space is sufficiently narrowed, the query is forwarded to SFS-WAIS gateways for processing. The relevant WAIS servers are determined based on the query and content labels. In our current prototype, there is one SFS-WAIS gateway per WAIS server to be queried. The search is done in parallel and the content router merges the results. The client retrieves documents directly from the remote information server thus bypassing the content router. For example:

```
=> Select  text:buddhism and text:tibetan
=> Expand
=> List-results
ANU-Asian-Religions:Bibliographical_reference.1
...
ANU-Asian-Religions:Bibliographical_reference.26
ANU-Shamanism-Studies:Samuel_Geoffrey_In_press_Ci.1
ANU-SocSci-Netlore:TIBETAN_FONTS_FOR_MACINTOSHES.1
Tantric-News:File_STS1089.1
Tantric-News:File_News_184.6
Tantric-News:File_News_387.4
Tantric-News:File_News_691.3
Tantric-News:File_News_792.2
Tantric-News:File_News_892.5
=> Retrieve ANU-Asian-Religions:Bibliographical_reference.26
TIBETAN SHAMANISM & BUDDHISM Bibliography
[This document last updated: 19 Feb 1992]
This is a rough draft, so please send additions
and emendations to Geoffrey Samuel - email:
sogbs@cc.newcastle.edu.au. These records are available
via WAIS and anonymous ftp from Coombspapers
at Coombs.anu.edu.au (150.203.76.2)
-------------------------------------------------------
Wylie, Turrell V. 1978. 'Reincarnation: a
political innovation in Tibetan Buddhism.'
Ligeti 1978: 579-586.
```

Figure 3 illustrates the prototype structure of the WAIS content router. CRS-WAIS and SFS-WAIS are

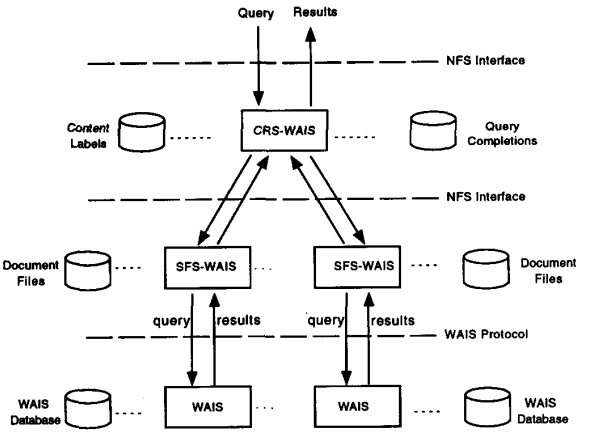


Figure 3: Structure of the WAIS content router

implemented as separate processes that can be run on the same or different machines. The client mounts CRS-WAIS as a standard NFS volume and SFS-WAIS servers are mounted client machines and on the machine where CRS-WAIS executes. CRS-WAIS consults the headline index to construct query completions and uses content labels to determine relevant servers. Then, pathname queries containing the identification of relevant servers are forwarded to SFS-WAIS processes. When all results are obtained, they are merged and returned to the client. Currently, the CRS-WAIS process runs on an SGI 4D/320S and SFS-WAIS processes on a Sparc Station 10.

## 5 Experience

We have used our prototype to locate and access documents on a large number of WAIS servers, and we gathered statistics on some example queries. The statistics are presented in Table 2 for simple conjunctions of terms. We report the number of relevant servers as determined by CRS, the number of documents found, the number of unreachable servers and the search latency for each query. We compare our CRS-WAIS router, which conducts searches in parallel, with a sequential search using **waisq**, a program in the WAIS distribution. In general, parallel searching performs better than sequential. However, there is no linear speedup in parallel searching: the latencies on different servers vary so that the parallel searching time is strongly limited by the slowest server. Some servers contain large databases, are accessed by many users, and have limited processing power. These servers limit the performance of any search. For example, the long search time for **infosystems** and **gopher**

| example query | number of servers | number of results | number of down servers | search time (parallel) | search time (sequential) |
|---|---|---|---|---|---|
| **buddhism and tibetan** | 5 | 71 | 0 | 396.0s | 752.2s |
| **distributed and synchronization** | 6 | 23 | 3 | 674.6s | 1442.1s |
| **infosystems** | 9 | 317 | 0 | 63.2s | 117.0s |
| **infosystems and gopher** | 9 | 132 | 1 | 680.5s | 743.7s |
| **multicast and broadcast** | 15 | 32 | 8 | 79.3s | 379.3s |
| **multimedia and authoring** | 12 | 47 | 1 | 108.8s | 196.9s |
| **poland and culture** | 11 | 11 | 0 | 47.6s | 494.7s |
| **video and quicktime** | 12 | 75 | 4 | 106.0s | 849.1s |
| **wireless and mobile** | 6 | 6 | 0 | 48.6s | 110.9s |

Table 2: Statistics on example queries

was due almost entirely to the processing time at one server (**biosci**) that continued to run long after other servers had finished. Moreover, one can experience 100% changes in search latency depending on whether the search is performed during the day or at night. Even if we take into account the delays at overloaded servers, the reported search times may appear long. However, we should bear in mind that the only alternative to using our content router is an exhaustive search of all servers, the operation which takes several hours to complete.

The appearance of efficient network searching servers like the WAIS content router may aggravate the load problems of some popular WAIS servers. As more users will search the increasing number of servers, the load of the servers will also increase and new solutions must be found to overcome this problem.

Availability of servers is also a problem. As can be seen from the table, there was at least one unreachable server for half of the sample queries. Our prototype creates virtual documents whose names signal the user that some server(s) did not respond (e.g. **comp.sys.sgi.misc.src:unreachable**). A better policy would be to keep track of unavailable servers, check periodically if they are up and eventually rerun a query. We are currently investigating adding this feature to our server.

We have also tested the performance of the query completion feature. Table 3 gives elapsed time for completions for a given term and the number of headlines that contain a given term. The figures show sufficient performance for interactive use of the feature.

Content routing based on WAIS catalog files has resulted in some unexpected behavior. For example, the query from Section 4 (**text:buddhism and text:tibetan**) was run on 5 servers and documents were found only on 4 servers (all servers were operational). A closer look at the catalog file of the missing server explained the problem. There is a headline with a pathname containing terms **buddhism and tibetan**, however the search on the WAIS server produced no result, because the file itself does not contain the terms:

**ftp.uu.net:doc/papers/coombspapers/otherarchives/electronic-buddhist-archives/buddhism-tibetan/**

| term | number of headlines | processing time |
|---|---|---|
| **buddhism** | 82 | 7.2s |
| **distributed** | 324 | 10.4s |
| **infosystems** | 87 | 6.2s |
| **multicast** | 126 | 7.1s |
| **multimedia** | 320 | 12.8s |
| **poland** | 123 | 11.1s |
| **video** | 349 | 11.0s |
| **wireless** | 48 | 4.9s |

Table 3: Query completion performance

Our experience with using the content router for locating and accessing WAIS servers shows that supporting boolean operations on servers is crucial to efficient searching. Only specialized terms can be efficiently used in our prototype, because general terms involve long searches with many result documents, even if a conjunction of two general terms returns a small result set.

While there is much work to be done, we were pleasantly surprised at the efficacy of content routing based on such simple content labels. With virtually no tuning based on high-level knowledge or sophisticated statistics, it was not difficult to explore a large set of information servers. In particular, we found query completion to be quite useful. Performance of the system is acceptable, though we would like to improve query performance and the handling of failures.

## 6 Conclusion

We have explored how the idea of content routing provides users with convenient associative access to documents in a large, distributed set of information servers. To be efficient and scalable, associative access to a large number of servers must involve only a restricted set of relevant servers. We have organized the search space in a network of WAIS servers using content labels constructed from the WAIS source and catalog files. Information-rich terms from document headlines are the key to identifying relevant servers. The scope of an initial query can be sufficiently narrowed and refined using a query completion feature so that the result set is efficiently computed.

We have built a prototype content router for WAIS servers. It is based on the SFS system and uses NFS as the underlying distributed protocol. We have successfully used the router to locate documents on a large number of WAIS servers. Along with demonstrating the feasibility of distributed finding in a large network of information servers, the prototype shows interactive performance limited primarily by the processing time at WAIS servers. The implementation of the content router for WAIS is the first attempt known to the authors of building a transparent, user-guided distributed information system on top of WAIS servers.

Future work will involve investigating more sophisticated techniques for constructing content labels, for organizing the network of servers, and for handling unreachable servers.

## Acknowledgments

We are grateful to our readers David K. Gifford, James O'Toole and Ron Weiss.

## References

[1] B. Alberti et al. The Internet Gopher protocol: A distributed document search and retrieval protocol. University of Minesota Microcomputer and Workstation Networks Center, Spring 1991. Revised Spring 1992.

[2] D. Barbara and C. Clifton. Information Brokers: Sharing knowledge in a heterogeneous distributed system. Technical Report MITL-TR-31-92, Matsushita Information Technology Laboratory, Princeton, NJ, October 1992.

[3] T. Berners-Lee et al. World-Wide Web: The information universe. *Electronic Networking*, 2(1):52–58, 1992.

[4] P. B. Danzig et al. Distributed indexing: A scalable mechanism for distributed information retrieval. Technical Report USC-TR 91-06, University of Southern California, Computer Science Department, 1991.

[5] A. Emtage and P. Deutsch. Archie – an electronic directory service for the Internet. In *USENIX Association Winter Conference Proceedings*, pages 93–110, San Francisco, January 1992.

[6] D. K. Gifford et al. Semantic file systems. In *Thirteenth ACM Symposium on Operating Systems Principles*. ACM, October 1991. Available as *Operating Systems Review* Volume 25, Number 5.

[7] Luis Gravano, Anthony Tomasic, and Héctor García-Molina. The efficacy of GLOSS for the text database discovery problem. Technical Report STAN-CS-TR-93-2, Stanford University, October 1993.

[8] B. Kahle and A. Medlar. An information system for corporate users: Wide Area Information Servers. Technical Report TMC-199, Thinking Machines, Inc., April 1991. Version 3.

[9] R. S. Marcus. An experimental comparison of the effectiveness of computers and humans as search intermediaries. *Journal of the American Society for Information Science*, 34(6):381–404, November 1983.

[10] R. S. Marcus. Advanced retrieval assistance for the DGIS gateway. Technical Report LIDS R-1958, MIT Laboratory for Information and Decision Systems, March 1990.

[11] J. Ordille and B. Miller. Distributed active catalogs and meta-data caching in descriptive name services. In *Proceedings of the 13th Internatinoal Conference on Dsitributed Computing Systems*, pages 120–129. IEEE, 1993.

[12] M. A. Sheldon et al. A content routing system for distributed information servers. In *Proc. Fourth International Conference on Extending Database Technology*, March 1994.

[13] P. Simpson and R. Alonso. Querying a network of autonomous databases. Technical Report CS-TR-202-89, Princeton University, Princeton, NJ, January 1989.

[14] R. M. Stein. Browsing through terabytes: Wide-area information servers open a new frontier in personal and corporate information services. *Byte*, pages 157–164, May 1991.