Content Routing for Distributed Information Servers

Mark A. Sheldon[1], Andrzej Duda[1,2], Ron Weiss[1]
James W. O'Toole, Jr.[1], David K. Gifford[1]

[1] Programming Systems Research Group, MIT Laboratory for Computer Science,
Cambridge, MA 02139, USA
[2] Bull-IMAG Systèmes, 38610 Gières, France

Abstract. We describe a system that provides query based associative access to the contents of distributed information servers. In typical distributed information systems there are so many objects that underconstrained queries can produce large result sets and extraordinary processing costs. To deal with this scaling problem we use *content labels* to permit users to learn about available resources and to formulate queries with adequate discriminatory power. We have implemented associative access to a distributed set of information servers in the *content routing system*. A content routing system is a hierarchy of servers called *content routers* that present a single query based image of a distributed information system. We have successfully used a prototype content routing system to locate documents on several user file systems and on a large number of information servers.

1 Introduction

The Internet contains over one million hosts that provide file service and other information servers specializing in topics such as news, technical reports, biology, geography, and politics. The Internet's vast collection of servers can be viewed as a distributed database containing a wealth of information. Unfortunately, this information is relatively inaccessible because there is no mechanism for browsing and searching it associatively.

The difficulty of providing associative access to a large number of distributed information servers lies primarily in problems of scale. The scale of the Internet, which is today only a fraction of its eventual size, is so great as to render infeasible any comprehensive indexing plan based on a single global index. In addition, the cost of distributing a query throughout the Internet is prohibitive. Finally, in very large scale systems, the number of results to a typical user query is incomprehensibly large. For these reasons, we expect that efficient associative access will require both *content routing* and *query refinement*. Content routing is the process of directing user queries to appropriate servers. Query refinement helps a user formulate meaningful queries. These query services can be implemented by a *content routing system* that will:

- Support a uniform query interface that allows progressive discovery of network contents and guides users in formulating queries of adequate discriminatory power.
- Propagate descriptions of server contents through a hierarchy of information servers.

- Route individual queries to available servers based on the expected relevance of the server to the query.

This paper explores the thesis that *content labels* can be used to organize the semantics of associative access to a distributed set of information servers, and that *content routers* based on the content labels can serve as the implementation basis of distributed associative access. We have implemented a prototype system that uses content labels to describe the contents of individual servers. The system contains content routers that provide associative access to underlying information servers. From the user's perspective, browsing and searching can be performed without regard to the location of individual data objects.

Using a prototype system, we have explored the automatic construction of content labels, tested their use in routing queries to information servers, and examined the basic performance characteristics of a multi-layered content routing system. We have used the system to locate documents on file systems and information servers. Initial experience suggests that a content routing system is an effective tool for exploring and searching a large network of information servers.

Unlike previous systems for locating and accessing information in a network, content routing provides a uniform, hierarchical framework combining associative access at two levels: associative access to sets of servers and associative access to documents on the servers. Important advances offered by our architecture include:

- **Usability:** Content labels can be used to provide feedback to users, enabling them to learn about available resources and formulate queries. Thus content labels organize the search space and enable navigation and discovery.
- **Scalability:** Content routing provides a uniform way to access large networks of information servers based on their contents. Layered content routing provides a scalable system architecture by maintaining limited knowledge of the attributes at the information servers.
- **Efficiency:** Users can efficiently search a network of servers using a content routing system. Content labels distill the content of information servers so that queries can be routed to appropriate servers.

In the remainder of this paper we review related work (Section 2), discuss the Content Routing System design (Section 3), discuss issues in constructing content labels (Section 4), describe the prototype implementation (Section 5), report on our experience (Section 6), and outline conclusions based on our experience (Section 7).

2 Related work

Previous work can be categorized as: distributed naming systems, network navigation systems and network-based information retrieval systems. No previous system combines a coherent architecture for associative access to distributed, heterogeneous information servers with query-specific feedback to help the user organize the search space.

Distributed naming systems such as X.500 [4], Profile [16], the Networked Resource Discovery Project [17], and Nomenclator [14] provide attribute-based access to a wide variety of objects; however, they do not support the hierarchical relationship between servers and data that our system achieves through its use of content labels.

Network navigation systems such as the Gopher system [1] and the World-Wide Web [3] provide a means for organizing servers to allow navigating among and browsing through remote information. Gopher provides the equivalent of a distributed file service, and World-Wide Web provides distributed hypertext documents. Discovering and locating information in these systems is difficult because these systems do not support query routing. Veronica [10] is a discovery system that maintains an index of document titles from Gopher menus.

Network-based information retrieval systems provide access to information on remote servers. The Wide Area Information Service (WAIS) [11, 19] provides a uniform means of access to information servers on a network. However, the user must choose appropriate servers from a directory of services. The Archie system [7] polls FTP sites on the Internet and indexes the files by name. A query yields a set of host/filename pairs which may be used to retrieve the relevant files manually. The Conit system [12, 13] provides a uniform user interface to a large number of databases accessible from several retrieval systems. There is no support for the automatic selection of relevant databases for a query. The Distributed Indexing mechanism [5] is based on precomputed indices of databases that summarize the holdings on particular topics of other databases. The architecture has a fixed three layer structure that is not flexible enough for content based access to heterogeneous information servers. Simpson and Alonso propose a querying architecture for a network of autonomous databases [18] that forwards user queries to known information sources. This approach has a probabilistic query semantics and does not support browsing with query refinement. The Information Brokers of the MITL Gold project [2] 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.

The concepts of query routing and query refinement were introduced in [9]. A query is routed to one or more databases that contains the result set of the query.

3 The Content Routing System

In this section we present the semantics and architecture of content routing systems. The overall job of a content routing system is to direct user requests to appropriate servers (content routing) and to help a user formulate meaningful queries (query refinement). Both of these tasks require a content routing system to employ detailed knowledge about what information is available and where it is located.

It is not feasible for a content routing system to maintain a detailed, single, global index of all information, nor is it practical to ask servers what they contain on a query-by-query basis. A single, detailed, global index would be quite large and costly to maintain in real time. Querying servers on a query-by-query basis is very expensive and would lead to substantial delays when there are thousands of servers.

Our system uses *content labels* as a compact and efficient means of describing server contents. As we will describe in detail, content labels allow a content routing system to implement both content routing and query refinement. The remainder of this section describes how the content routing system organizes the search space of a distributed information system (Section 3.1), provides a detailed description of the system operations that comprise the system's interface and semantics (Section 3.2), and illustrates the intended use of the system through examples (Section 3.3).

3.1 System Organization

A content routing system extends a traditional information retrieval system to a distributed and hierarchical network of information servers. It supports conventional document types, such as text, electronic mail, and video, along with conventional operations on those documents. A content routing system defines a new document type, called a *collection*, that supports new operations designed to enable users to browse and search a large distributed information system.

A *collection* document consists of a set of documents (possibly including other collection documents) together with a description of the set called a *content label.* Because a collection document contains other documents including collections, it can be viewed as a hierarchical arrangement of documents in a directed acyclic graph (see Figure 1). A content label is a compact representation of the set of member documents in the form of a query predicate. All the documents of a collection must satisfy the content label of that collection interpreted as a predicate.

A content routing system is a network of information servers that provides a distributed search space organized by content labels. Leaf nodes in the network are endpoint information servers that store conventional documents. The hierarchical network of internal nodes is composed of *content routers*, which are information servers that support collection documents. The structure of the network mirrors the collection hierarchy of Figure 1, where the server underlying each collection stores the conventional documents and content labels of the layer below.

A content router implements a collection as an information retrieval service (which in general runs remotely) and a content label. User query and retrieval operations can be applied to content labels or routed to the servers that implement the collections. If an operation is forwarded to multiple servers then the results are merged. A content router may elect to forward operations to servers or to refuse to do so depending on the cost effectiveness or expense of that operation.

An information server registers with content routers by providing a content label and an interface for the content router system operations. Our design does not limit the form of content labels. In principle, a content label could consist of the disjunction of all of the attributes that appear in a collection. Alternatively, a content label could simply be the name of the host that contains a set of files. In practice, we expect content labels to lie between these two extremes, containing attributes including host names, domains, authors, libraries, priorities, and subjects. Pragmatic issues in the construction of content labels are explored in Section 4.

The system operations interface includes operations for query, document retrieval, and query refinement requests (see section 3.2). Servers may be organized by administrative domain (e.g., all the servers at MIT), by a subject hierarchy (e.g., based on the Library of Congress categorization scheme), or by a mechanism in which servers

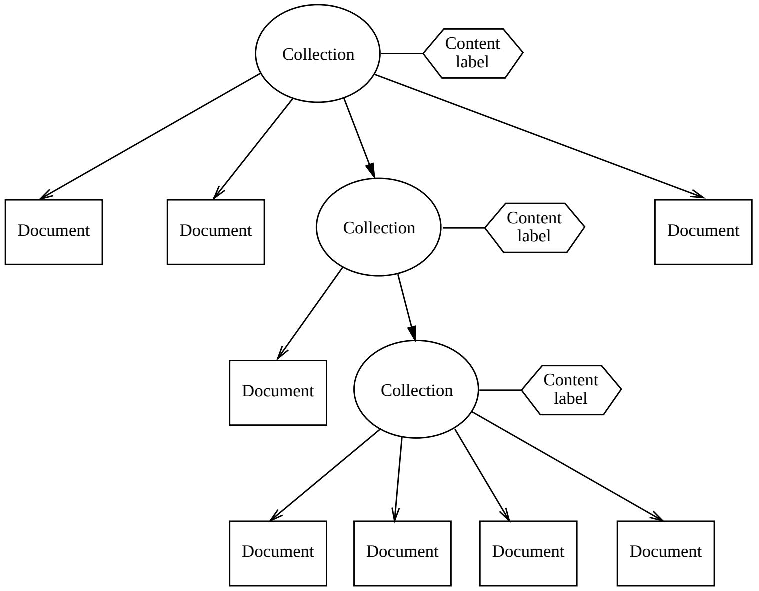

Fig. 1. A content routing system provides access to a hierarchy of collections.

target particular information markets. To support query refinement, a content router may extract term relationship information from its registered collections.

3.2 System Operations

A content routing system supports operations for query, document retrieval, and query refinement, in addition to conventional document operations. A user may *select* a set of documents from the collection using a query, *list results* of a selection, and *retrieve* an individual document. For collection documents, the system interface also supports the new operations *show fields*, *show values*, *expand*, and *search*, which are described below. Together, these services are designed for browsing and searching large distributed information systems.

For the purposes of this paper and our prototype system, we have chosen a simple predicate data model. *Documents*, which may be text files, directories, digital video or any other data objects, are associated with sets of *attributes*. Each attribute consists of a *field* name and a *value*. Typical attributes are `text:database` and `author:ullman`. Some field names have predefined semantics, e.g., `author`, and may or may not make use of a controlled vocabulary. Other fields may have semantics that vary from domain to domain, e.g., `category`. Individual documents may also export their own field names, thus the set of fields is extensible. Values can assume a variety of types, including integers and dates. In our current prototype however, values are strings.

A *query* is a boolean combination of attributes. When a query is applied to a set of documents, the subset of documents that satisfies the query is called a *query result set*.

A collection document, consisting of a set of documents (including other collections) and a content label, helps to organize the search space for the user. Because the content label accurately characterizes the contents of the collection, it can be used by the system to guide search operations. For example the following very small content label implies that every document of the collection has attribute `subject:database` and `collection-type:software` and is either in the `cmu.edu` or the `stanford.edu` domains.

```
[ (subject:database) and ((domain:cmu.edu) or (domain:stanford.edu))
                     and (collection-type:software) ]
```

The basic operations of the content routing system are as follows:

The *open* operation initializes a connection to a particular collection, allowing the user to search and browse its contents. The open operation initializes the current result set to be the entire contents of the collection.

The *select* operation reduces the current result set to the subset of documents that matches the given query. A collection matches a query if its content label matches the query.

The *retrieve* operation allows the user to view and save the document. A retrieval request applied to a collection document produces a human-readable description of the contents of that collection.

The *search* operation allows the user to submit a query to the content routing system for query routing. The routing system automatically forwards the query to relevant information servers. To a user specifying search queries, the system appears as a single large virtual information server. This virtual information server contains all the conventional documents that are reachable from the currently open collection.

The *expand* operation allows the user to access the contents of a collection or set of collections. When a collection from a query result set is expanded, the query is applied to the components of the collection and matching components are added to the result set. The expansion operation does not affect a conventional document. Expanding a set of documents results in the union of the expansion of every document in the set. The resulting documents may then be queried, retrieved, or expanded. When a result set has been expanded, the user has access to new query terms, namely, those attributes arising from the component documents of the collections.

While composing a query, a user can learn about possible terms by using the *query refinement* feature. The user can learn from the system the set of available fields, and the system will also suggest potentially interesting values for fields based on the query fragment the user has presented.

The *show fields* operation displays the set of attribute field names known to the currently open collection. Ideally, the set of fields displayed should be limited to those found in the documents of the current result set.

The *show values* operation suggests information rich values for a given field that are related to the documents in the current result set. Our design leaves the precise behavior of this operation as an implementation issue.

The *list results* operation enumerates the documents in the current result set.

These operations for guiding query formulation help the user manage the complexity of the search space. A user browses documents by alternately broadening and narrowing queries. Typically users start with broad queries. The set of documents to peruse becomes smaller as a query is refined. The user refines queries either by using system recommended completions or by using attributes discovered in interesting content labels. When a query is sufficiently narrowed, its collection documents may be expanded. This process continues in a pattern of contracting selections alternating with expansions.

3.3 An Example

This section contains a simple, extended example of how a content routing system might work, and in fact comes from our running implementation. However, these examples are illustrative and are not intended to limit future implementations. For pedagogical reasons, we use a simple textual interface, although our system provides a graphical browser.

The user can perform a search that will automatically retrieve relevant documents from the information servers in the system. First, the user opens a collection of WAIS collections, lists the fields available for query construction, and defines a query. The user then refines the query by inquiring about possible query completion values, and then incorporates the system's suggestions into a new query. Finally, the user displays the result set that matches the refined query, and retrieves a particular document.

```
=> open-collection  crs-wais
=> show-fields
   administrator:      field:          label:      port:
   collection-type:    hostaddress:    location:   text:
   cost:               hostname:       owner:
=> search  text:buddhism
=> show-values  text:
   academic     chinese      data       religion     society
   asian        cultural     omni       religions    tibetan
   bases        culture      raw        resource     zen
=> search  text:buddhism and text:tibetan
=> 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
=> retrieve  ANU-Asian-Religions:Bibliographical_reference.26
   TIBETAN SHAMANISM & BUDDHISM Bibliography [Last updated: 19 Feb 1992]
   ---------------------------------------------------------------------
   Wylie, Turrell V. 1978. `Reincarnation: a political innovation in
   Tibetan Buddhism.' Ligeti 1978: 579-586.
```

Alternatively, the user can browse the system and progressively explore the network resources. In the following example, the user opens a collection, selects all the collection documents that contain a specific attribute, and displays the content label for one of these collections.

```
=> open-collection crs-wais
=> select  text:buddhism
=> list-results
   ANU-Asian-Religions      ANU-Thai-Yunnan              lists
   ANU-Shamanism-Studies    ANU-ZenBuddhism-Calendar     mailing-lists
   ANU-SocSci-Netlore       file-archive-uunet           Tantric-News
=> retrieve  Tantric-News
   (hostname: sunsite.unc.edu) and (hostaddress: 152.2.22.81) and
   (collection: Tantric-News) and (label: server) and (label: WAIS) and
   ...
   (text: Society) and (text: Tantric) and (text: Studies)
   ...
```

When the user discovers collections that may contain relevant information, the user can `expand` the current query to retrieve matching documents from these collections.

```
=> select  text:buddhism and text:tibetan
=> list-results
   ANU-Asian-Religions     ANU-SocSci-Netlore    Tantric-News
   ANU-Shamanism-Studies   file-archive-uunet
=> 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
```

Note that after the `expand` operation above, the user will get the set of fields defined in the selected collections. Since all the documents in the expanded result set above were collections, the fields from the original collection may no longer be available.

```
=> show-fields
   author:     date:       imports:    owner:       subject:   title:
   category:   exports:    name:       priority:    text:      type:
```

4 Constructing Content Labels

A good content label will contain two kinds of attributes:

- Administratively determined *value added* attributes that describe a collection but may or may not appear in the documents themselves. For example, an administrator may want to advertise that a server has the attribute `collection-type:software`.
- Attributes automatically derived from the collection contents, possibly using various statistical tools. For example, frequently occurring descriptive terms are good candidates for content labels.

If desired, it is possible to enforce a uniform meaning for a given field name across servers and document types. For example, `owner` could always refer to the owner of a document. It is also possible to allow the meaning of field names to be defined differently for different servers or document types.

Content labels represent a compromise for a scalable architecture, since there are practical limits on the size of content labels. Limiting content label size not only helps manage local resources, but also encourages administrators of remote information servers to make effective use of their content label advertising budget. A budget scheme will give priority to specialized servers when users submit more specific queries.

Server	*# of attributes*	*Server size (MB)*	*Index size (MB)*
comp	564493	65	50.5
rec	309473	43	29.5
users1	247914	184	29.5
nyt	165678	174	29.3
users2	99451	28	15.6
ap	36620	29	3.9
total	1423629	403	158.3
unique	1077522		

Table 1. Information servers statistics

In order to explore how content labels can be generated automatically for large collections of objects, we have gathered some attribute statistics on six information servers. Table 1 gives the characteristics of the servers: the number of distinct attributes, the server size (total size of documents in the database in megabytes), and the index size (size of the index data structures). *Comp* is the USENET `comp` newsgroup hierarchy, *rec* is the `rec` newsgroup hierarchy, *users1* and *users2* are two user file systems, *nyt* is a database of New York Times articles, and *ap* is a database of Associated Press articles.

Figure 2 plots the total number of unique attributes versus the number of servers. It shows that `text` comprises a majority of the attributes and that the number of distinct attributes is large. A content label containing all attributes (or even all `text` attributes) would be infeasible. However, some information-rich attributes such as `author`, `title` and `subject` have a small set of information-rich values and are thus good candidates for content labels.

Content labels must reconcile two conflicting objectives. On one hand, they must contain terms that characterize the collection as a whole (typically high frequency terms), and on the other hand, they must be able to distinguish servers with respect to each other. To evaluate the discriminatory power of attributes, we gathered data on the distribution of attributes over servers. Figure 3 shows for a given number of servers how many attributes appear only on that many servers. For example, there were 10 `category` attributes that appeared on three servers. If a given attribute has a narrow distribution, that is, it identifies a small number of servers, then it can used for routing. Wide distribution terms are useful for describing collections for browsing. As shown in the figure, low frequency attributes such as `owner`, `category`, and `type`

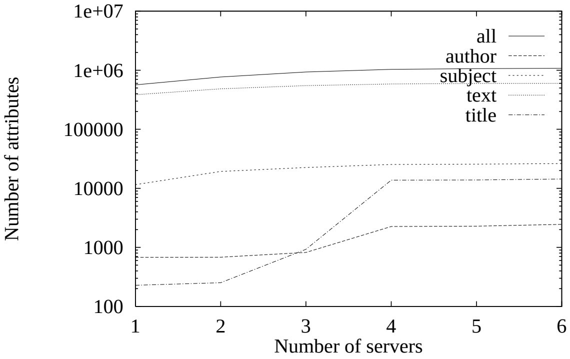

Fig. 2. Cumulative number of attributes

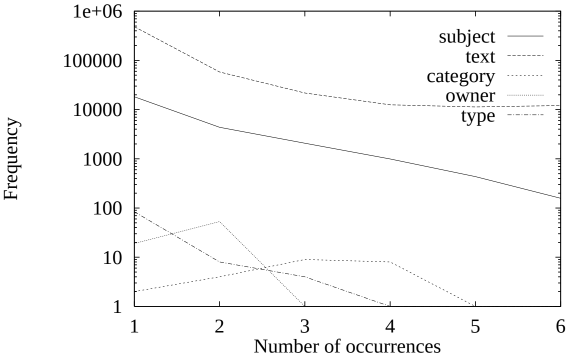

Fig. 3. Attribute histogram

have discriminatory power over servers and can be used for content routing. Higher frequency attribute fields like `text` and `subject` are more common so that the most frequent terms can be used for categorizing the collection as a whole and propagated to upper routing layers.

5 Implementation

We have built a prototype implementation of a content routing system that provides query routing to an extensible number of servers. The system uses the Semantic File System interface [8] for both information servers and content routers. Because it is implemented as an SFS server, our content router is a user level NFS server that may be mounted by any NFS client. Figure 4 shows the architecture of our prototype content router implementation.

Path names that pass through the NFS interface to the server are interpreted as queries and results are returned in dynamically created *virtual directories*. On demand, the server computes the contents of virtual directories. Queries at the content router apply to the content labels for the collections registered there. The content router uses the query to determine the set of collections. Our implementation constructs suggested refinements for a given query by identifying terms frequently collocated with the terms appearing in the query.

The server performs an expand operation by forwarding the query (and subsequent query refinements) to the set of servers whose content labels match the query. The merged results are presented to the user. Our current implementation of the search operation uses syntactic hints embedded in queries. These hints indicate to the router which terms should apply to content labels. We use these hints because we are still experimenting with various routing algorithms.

Our content router interposes itself between a client and an information server during query processing when results from more than one server must be combined.

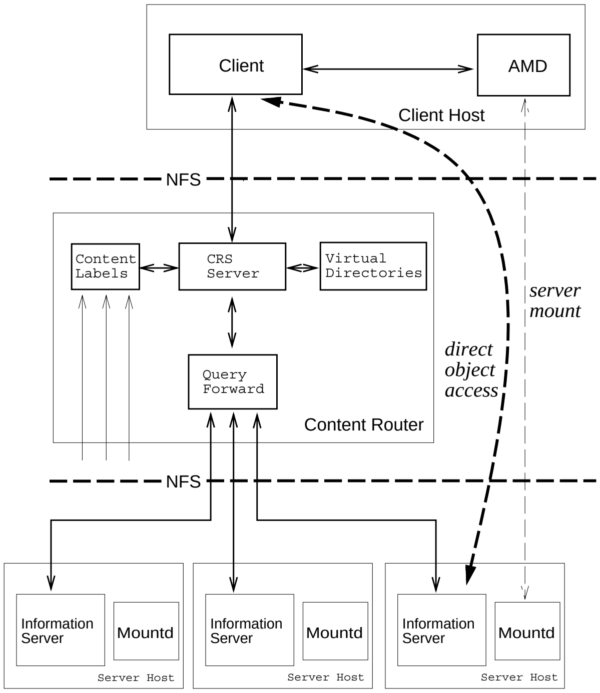

Fig. 4. A prototype content router.

We call this approach *mediated processing* because the content router mediates the responses from multiple information servers to present a collective result.

When only a single server is sent a query or once a document of interest is identified, a client is instructed to bypass the content router and contact the end-point server directly. We call this approach *direct processing.* Direct processing allows a client to obtain maximum performance from the end-point server and also minimizes content router load. In our architecture, client to server forwarding is implemented by having the content router return a symbolic link to a client in response to a request for a virtual directory or file. The symbolic link returned refers to the end-point server and is dynamically resolved by the client with the help of an automount daemon [15].

The content router collects content labels from a set of servers to which it will provide access. We have experimented with two ways of creating content labels: for our SFS file systems, content labels are administratively determined and for WAIS servers content labels consist of disjunctions of attributes obtained from the individual WAIS source and catalog files. The WAIS catalog files contain headlines for each document (subject of a message or title of an article). Thus, they contain information-rich terms that characterize document content fairly well. From 492 existing WAIS servers, we have retrieved 371 catalog files which occupy 352.5 megabytes (MB) of disk space. As only part of catalog files (headlines) is used for constructing content labels and only unique terms are kept, the size of the resulting content labels is 12.4MB.

To provide access to WAIS servers, we implemented an SFS-WAIS gateway so that WAIS servers are viewed as SFS servers [6]. The SFS-WAIS gateway translates SFS queries into WAIS questions and uses the public domain client WAIS code for querying WAIS servers.

6 Experience

We have used several configurations of our content routing system. We measured the performance of query routing in a small system of four semantic file systems. We measured the performance of a single query router providing access to 492 WAIS servers located around the world. We also verified that direct processing achieve better performance than mediated processing in a multiple-layered system.

Table 2 shows representative performance of a content router. We expect that we will be able to further lower query processing times on our system because our prototype implementation is not tuned. The information servers run on an SGI 4D/320S and a heavily loaded DEC Microvax 3500. The content routers and clients were run on Sparc Station IPX's. All these machines were interconnected with a 10Mbit/s Ethernet. These tests did not process queries on multiple servers in parallel.

Example query	*Number of servers*	*Number of results*	*Search time (sequential)*
`library:users and owner:`	1	88	1.2s
`library:users and text:semantic and owner:sheldon`	1	22	0.6s
`library:users and text:semantic and text:library and owner:sheldon`	1	9	1.2s
`location:mit and extension:video`	2	60	2.6s
`location:mit and owner:gifford`	2	31	1.4s
`text:toys and text:williams`	4	27	9.9s

Table 2. Routed query performance on four local SFS servers

We also used our prototype to locate and access documents on a collection of 492 WAIS servers, and we gathered statistics on some example queries. The server in this case ran on an SGI 4D/320S, and the WAIS servers were distributed throughout the internet and ran on unknown types of machines. Since many public WAIS servers do not support conjunction, our SFS-WAIS gateway is forced to run a separate remote query for each term and compute result set intersections locally. This greatly increases processing times and encourages users to restrict themselves to highly specialized query terms. The statistics for some sample queries are presented in Table 3. We compare the content router conducting searches on different WAIS servers in parallel with sequential search using `waisq`, a program in the WAIS distribution. The `waisq` program was given the servers chosen by our content router. In general, parallel searching performs better than sequential. However, there is no linear speedup: latencies on different servers

Example query *(all attributes are* `text`*)*	*Number of* *servers*	*Number of* *results*	*Number of* *down servers*	*Search time* *(parallel)*	*Search time* *(sequential)*
`bosnia and clinton`	4	86	0	219.0s	1003.4s
`buddhism and tibetan`	5	71	0	397.7s	646.6s
`multimedia and maestro`	13	23	0	215.2s	233.3s
`perfect and hashing`	8	31	0	52.7s	147.6s
`transactions and nested`	7	13	1	46.5s	151.4s

Table 3. Example query performance routing to 492 WAIS servers

vary so that parallel search 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 **`buddhism`** and **`tibetan`** was due almost entirely to the processing time at one server (**`ANU-Asian-Religions`**) that continued to run long after other servers had finished.

From our experience, we believe that query routing in a hierarchical collection of distributed servers is feasible. Demanding applications like digital video can achieve adequate throughput because content routers do not mediate object accesses. 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. However, we were pleasantly surprised at the efficacy of routing based on content labels derived from WAIS catalogs. 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.

7 Conclusions

We have designed and implemented a content routing system that allows users to browse, locate, and access documents of interest in large, heterogeneous, distributed information systems. We have organized the search space using content labels describing 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 based on the semantic file system model. We have successfully used our system to locate data objects on servers in our laboratory and to locate documents on a large number of WAIS servers. Along with demonstrating the feasibility of our approach, the prototype shows interactive performance mainly limited by the processing time at end-point information servers. The most surprising result was that even with relatively poor information about remote server contents, content labels were quite useful for browsing and locating interesting information.

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

References

1. B. Alberti, F. Anklesaria, P. Linkner, M. McCahill, and D. Torrey. 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, Oct. 1992.
3. T. Berners-Lee, R. Cailliau, J.-F. Groff, and B. Pollermann. World-Wide Web: The information universe. *Electronic Networking*, 2(1):52–58, 1992.
4. CCITT. The Directory - Overview of Concepts, Models and Services. Recommendation X.500, 1988.
5. P. B. Danzig, S.-H. Li, and K. Obraczka. Distributed indexing of autonomous internet services. *Computing Systems*, 5(4), 1992.
6. A. Duda and M. A. Sheldon. Content routing in networks of WAIS servers. Submitted for publication, Sept. 1993.
7. A. Emtage and P. Deutsch. Archie – an electronic directory service for the Internet. In *USENIX Association Winter Conference Proceedings*, pages 93–110, San Francisco, Jan. 1992.
8. D. K. Gifford, P. Jouvelot, M. A. Sheldon, and J. W. O'Toole. Semantic file systems. In *Thirteenth ACM Symposium on Operating Systems Principles.* ACM, Oct. 1991. Available as *Operating Systems Review* Volume 25, Number 5.
9. D. K. Gifford, J. M. Lucassen, and S. T. Berlin. An architecture for large scale information systems. In *10th Symposium on Operating System Principles*, pages 161–170. ACM, Dec. 1985.
10. H. Hahn and R. Stout. *The Internet Complete Reference.* Osborne McGraw-Hill, Berkeley, California, 1994.
11. B. Kahle and A. Medlar. An information system for corporate users: Wide Area Information Servers. Technical Report TMC-199, Thinking Machines, Inc., Apr. 1991. Version 3.
12. 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, Nov. 1983.
13. R. S. Marcus. Advanced retrieval assistance for the DGIS gateway. Technical Report LIDS R-1958, MIT Laboratory for Information and Decision Systems, Mar. 1990.
14. 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.
15. J.-S. Pendry and N. Williams. Amd: The 4.4 BSD automounter reference manual, Mar. 1991. Documentation for software revision 5.3 Alpha.
16. L. Peterson. The Profile Naming Service. *ACM Transactions on Computer Systems*, 6(4):341–364, Nov. 1988.
17. M. F. Schwartz. The Networked Resource Discovery Project. In *Proceedings of the IFIP XI World Congress*, pages 827–832. IFIP, Aug. 1989.
18. P. Simpson and R. Alonso. Querying a network of autonomous databases. Technical Report CS-TR-202-89, Princeton University, Princeton, NJ, Jan. 1989.
19. 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.