A Content Routing System for Distributed Information Servers

Mark A. Sheldon

Andrzej Duda[†]

Ron Weiss

James W. O'Toole, Jr.

David K. Gifford

June 1993

MIT/LCS/TR-578

Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139

[†]Also with Bull-IMAG Systèmes, and INRIA

This research was sponsored by the Defense Advanced Research Projects Agency (DARPA) under contract DABT63-92-C-0012.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

Keywords: distributed information retrieval, query routing, query processing, content-based access, distributed heterogeneous databases

Abstract

We describe the first system that provides query based associative access to the contents of distributed information servers. Queries describe desired object *attributes*, and are automatically forwarded to servers that contain relevant information. 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 quickly formulate queries with adequate discriminatory power. We present experimental data that show that certain content label attributes can be automatically chosen. We have implemented associative access to a distributed set of information servers in the *content routing system*. 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. Experiments motivated by our video access service show that substantial performance benefits result when content routers are removed from the client-server path once an object of interest is found.

1 Introduction

We estimate that there are over 2^{20} hosts that provide file service in the Internet, with an average of 2^{16} files per host. This collection of files represents a vast potential resource. However, the lack of associative access to this collection makes it difficult to discover relevant information.

Even when implementation issues are ignored, it is difficult to formulate a query language and corresponding schema that allow a user to easily locate objects of interest in a collection of 2^{36} objects. For example, if a user is willing to examine a result set of at most 2^8 objects, then the user's query must contain at least 28 bits of information. Thus the semantic challenge of query based associative access is to provide a convenient way for users to formulate queries that contain sufficient discriminatory power. Once the semantic problems are solved there remains the practical problem of how to implement the search process.

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 upon the content labels can serve as the implementation basis of distributed associative access. We call a collection of information servers that implements associative access with content labels and content routers a *content routing system*. We have implemented a content routing system that provides associative access to its underlying servers without regard to where files are located.

A content label is a succinct description of a *collection* of objects. Content labels can include administratively determined attributes, such as library names, group names, logical locations, and other assigned properties and relationships designed to aid users. Content labels can also include automatically extracted attributes, such as Internet domains, the authors of objects, object types, path names, subjects, key words from abstracts, and so forth. A collection of objects can correspond to the contents of a host, the contents of a removable disk, a set of collections, or other groupings of objects.

The semantics of the query language is designed to help a user formulate a query that reasonably bounds the object search space. If user queries are not appropriately constrained, then query processing consists of two extreme cases: either queries are broadcast to all information servers or each content router must have complete knowledge of all object attributes. Neither of these approaches scales well. Query routing based on content labels represents a compromise for a scalable architecture. In our approach, a user must initially state a query that includes attributes included in content labels. Available attributes can be enumerated to enable query assembly from prespecified components. Once the search space is narrowed, more attributes become available to the user. When the query contains enough information all attributes are available for query construction.

Query routing is used for both query formulation and query processing. A content router may require that a query be sufficiently narrow to describe a router specific number of collections before the router forwards a query to its underlying collections. From the user's perspective a router provides abstracts of its available collections in terms of content labels. Once a query is specific enough, the router forwards the query to its underling collections, and the user can employ the full range of attributes that had been abstracted previously from the specified collections.

One way to associate attributes with objects in a collection is to use *transducers* [GJSO91] that automatically extract attributes from files, directories, servers, and other objects. Transducers are object-type specific, and serve to convert an unstructured object into a set of descriptive attributes. A transducer is automatically run on an object when it is created or updated. Default transducers exist for text files, source files, object files, mail files, directories, and many other popular object types. Users can extend the semantics of the system with customized transducers.

In a large content routing system a user begins at a top level router. At a top-level router a user constructs a query based on high level attributes such as author, library name, Internet domain, or host name. Once a query is narrowed, the top level router forwards the query to matching collections. The user can then elaborate the query with full-text words, procedure names, file extensions, or other common attributes. Ultimately, a set of matching objects is returned in response to a suitably constrained query. If a user knows enough to provide a suitably constrained query to a top-level router, the query will be automatically forwarded to appropriate collections and a result set will be produced (see Figure 1).

Content routers also serve an administrative role in the management of a distributed system. Typically, a content router will service an organizational unit. These units can control access to the servers within the unit domain, and can also provide value added information exporting the unit logical attributes. Thus, the units affect the logical view of the network underneath them.

We have implemented a prototype content routing system to explore how content labels can be used for query routing and the performance im-

Figure 1: Associative access to video segments through a content routing server.

plications of using multiple layers of query routing. Figure 1 shows a video browser that uses the query facilities of our content routing system to locate video segments of interest on local servers. We chose to experiment with video access because of its demanding bandwidth and real-time requirements.

In the remainder of this paper we discuss the previous work upon which we build (Section 2), the semantics of queries and content labels (Section 3), experimental data from our implementation of a content routing system (Section 4) and conclusions based upon our experience (Section 5).

2 Related work

Previous work can be broken down into the following broad categories: distributed naming systems, network navigation systems and network-based information retrieval systems.

Distributed naming systems such as X.500 [CCI88], Profile [Pet88], and the Networked Resource Discovery Project [Sch89] provide attribute-based access to a wide variety of objects; however, they do not support the hierarchical relationship between servers and data which our system achieves through its use of content labels. Also, the systems are not integrated into a file system, nor do they provide automatic attribute-based access to the contents of objects.

Network navigation systems such as the Gopher system [AAL+91] and the Word-Wide Web [BLCGP92] provide a means for organizing servers into a structure that allows navigating among and browsing through remote information on servers. 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. However, it would be possible to install a content routing system into either system. Fremont [WCS93] is a research prototype for discovering key network characteristics, such as hosts, gateways and topology.

Network-based information retrieval systems provide access to information on remote servers. The Wide Area Information Service (WAIS) [KM91, Ste91] provides a uniform means of access to information servers on a network. WAIS permits information at remote sites to be queried, but relies upon the user to choose an appropriate remote host from a directory of services. The Archie system [ED92] 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 Alex file system [Cat92] is an NFS-FTP protocol converter. It integrates FTP servers into the local file system, but does not address the issue of associative access to remote files. The Conit system [Mar83, Mar90] 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 users query. The Distributed Indexing mechanism [DANO91] 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 which is suitable for incorporation of bibliographic databases, but it is not flexible enough for content based access to file servers. Simpson and Alonso propose a querying architecture for a network of autonomous databases [SA89] that forwards user queries to known information sources. The knowledge about existing sources is acquired by exchanging messages with neighboring hosts. This approach has a probabilistic query semantics because finding all relevant objects is not guaranteed. The Information Brokers of the MITL Gold project [BC92] 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.

Unlike the previous systems, our architecture combines in a transparent way associative access to distributed information servers with content based access to objects on the servers. Key advances offered by the present work include:

- *Scalability:* Query routing provides a uniform means of content-based access to large networks of information servers. Query routing further provides a mechanism for routing updates to relevant information servers. It uses a layered routing architecture that maintains limited knowledge of the attributes at the information servers which is essential for scaling.
- *Usability:* Content labels provide a distillation of information server content so that user queries can be effectively routed to appropriate servers. Content labels also permit users to learn about available resources and to quickly formulate queries with adequate discriminatory power. The object search space is organized using content labels, and enables navigation and discovery through a large space of servers such that the scope of an initial query can be sufficiently narrowed and refined.
- *Efficiency:* location of objects in a hierarchical network of servers is complemented by direct access to end-point servers and data objects once an individual server and/or object is identified.

3 Semantics

Our goal is to design a distributed information system architecture for content-based navigation and associative access to data objects. Users locate desired files, directories, or other objects through use of queries formulated in terms of content attributes. The fundamental requirement is to allow users to describe conveniently the desired objects in the distributed information system, while simultaneously excluding undesirable objects.

Our architecture is composed of two conceptual layers: the query routing layer, and the end-point information server layer (see Figure 2). Internally, the routing layer is configured as a hierarchical system of nodes, called *content routers.* (See Figure 3.) These are in charge of routing user queries to other content routers or to end-point information servers.

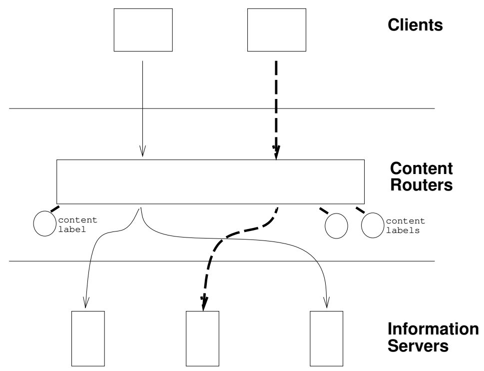

Figure 2: Routing queries to information servers.

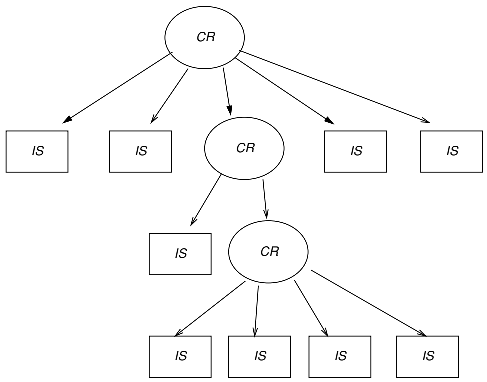

Figure 3: A content routing system provides access to a network of information servers.

3.1 Query and content label semantics

A query is a boolean combination of attributes. Each attribute consists of a *field* name and a *value*. Typical attributes are `text:multiprocessor` and `author:bershad`. When a query is applied to a collection of objects the subset of the collection that satisfies the query is called a *query result set*. We chose a perfect search scheme that guarantees that a result set contains all objects that match a query on accessible servers in the system. 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 that abstracts the contents of a collection. For example, a 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 member of the collection has attribute `collection-type:software` and `subject:operating-systems` and is either in the `cmu.edu` or the `standford.edu` domains.

In typical distributed systems there are so many objects that underconstrained queries can produce large result sets and extraordinary processing costs. There are several methods that can be used individually or in combination to limit the object search space to permit scaling:

- The user can supply the physical location of the server that holds the data. For example, in the WAIS system, which contains information about hundreds of servers, the user must choose the physical location to search.
- The user can supply a query that contains *value added* attributes that are created by some administrative means. For example, `collection-type:software` might restrict the search space to objects that reside in software repositories.
- The user can supply information-rich natural attributes. For example, if the attribute `owner:clinton` occurs in only five servers, a query that contains this attribute can be effectively routed to the appropriate servers. A *natural* attribute (as opposed to a value added attribute) is one that actually arises from indexing an object and thus is emitted by an object transducer.

The content label of a collection must correspond to the logical disjunction of the attributes of all collection member objects. Aside from this constraint, our architecture does not limit the form of content labels. In principle we could have a content label that consisted of the disjunction of all of the words that appeared in a collection. Alternatively, a content label could simply be the name of the host that contains a collection 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.

The lack of constraints on the form of content labels and the range of attribute field names and values makes our architecture extensible. If desired, it is possible to enforce a uniform meaning for a given field name across object transducers. For example, `owner:` could always refer to the owner of an object. It is also possible to allow the meaning of field names to be defined by each transducer. This latter requirement may necessitate including object attributes that uniquely identify the semantics of the transducer used on the object.

Content routing is performed by organizing content labels into a collection. When a query implies a content label, the content label is said to match the query. The set of collections relevant to a query is precisely defined by the corresponding content labels that match the query. If the set of matching collections is too large, the user can refine the query based upon the attributes present in the matching content labels.

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 the query. The user iteratively refines the query until enough discriminating power is present. At that point, the search space is sufficiently limited, and the user may submit unrestricted queries that are processed at all relevant servers.

3.2 An example

The user enters the content routing system

```
% cd /crs
```

To get started, we can find out what fields are supported for navigating among servers

```
% ls crs-field:
administrator:      hostname:
cost:               immutable-field:
collection-type:    label:
crs-field:          location:
```

```
field:              owner:
hostaddress:        port:
```
The user may then enumerate values of interesting fields.

```
% ls location:
massachusetts       new_england
mit                 united_states
% ls collection-type:
articles       files          user
digital        news           video
```
The user may explore an interesting attribute. There are two user file system servers available, Their associated content labels are available for browsing (the `.desc` *files)*

```
% cd collection-type:/user
% ls
users1          users1.desc
users2          users2.desc
```
The user wants to locate some interesting objects within that search space. The current implementation requires an explicit indication to perform the search.

```
% cd object-search
```
Find all files owned by **jones** *that contain the text keywords* **content** *and* **routing***. The system will automatically merge results from the two servers discovered previously.*

```
% cd owner:/jones/text:/content/text:/routing
% ls
implementation.tex      related-work.tex
paper.text              sosp91.tex
proposal-body.tex       slides.tex
prop.tex                tour.text
psrg-mail.text
```
Find all movie previews

```
% cd /crs/collection-type:/video
% cd object-search/subject:preview/ext:/movie
% ls
bond1.movie             father_of_bride.movie
forever_young.movie     toys.movie
```

IS	*# of attributes*	*IS 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

3.3 Attribute statistics

In order to demonstrate that certain attributes with high discriminatory power can be automatically discovered and used in content labels, we have gathered attribute statistics on six information servers. Table 1 gives the characteristics of the servers: the number of distinct attributes (field-value pairs), the server size, and the index size. *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 4 shows how the total number of unique attributes grows when more servers are added. It can be seen from this figure that `text:` comprises a majority of the attributes. Due to the sheer number of distinct attributes, a system that maintains complete knowledge about all `text:` attribute values for content labels appears impractical. However, because there are only 1000 distinct `author:` attributes, they might be used for content labels.

To evaluate the discriminatory power of attributes over servers, we gathered histograms of the occurrence of attributes on servers. The histograms give the number of attributes that occur on $1, 2, \ldots, 6$ servers. These statistics are helpful in deciding whether query routing can be done based on certain attribute fields. If a given attribute has a narrow distribution, that is it identifies a small number of servers, then its field can be propagated to content routers in a content label. Figures 5, 6 and 7 present the histograms for low, medium and high frequency attributes. Low frequency attributes such as `owner:` `category:` and `type:` as well as some medium frequency attributes such as `newsgroups:`, `date:` and `organization:` have discriminatory power over servers and can be used for content label routing. Our

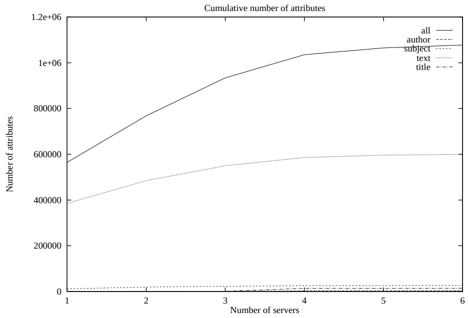

Figure 4: Cumulative number of attributes in n servers

implementation has propagated these attributes into content labels for the six information servers that we used. Higher frequency attribute fields like `text:` and `subject:` obviously cannot be exploited in the same way.

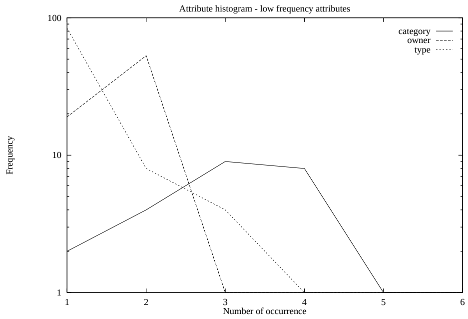

Figure 5: Attribute Frequency Histogram for Low Frequency Attributes

For `text:` attribute we have further analyzed the gathered statistics in order to find out whether we can group some infrequent values in a new attribute that can be propagated in a content label. The results are encouraging and we are continuing to investigate techniques for automatically determining what attributes are useful for routing.

4 Implementation

We have built a prototype implementation of a content routing system that provides query routing to an extensible number of servers. Server brows-

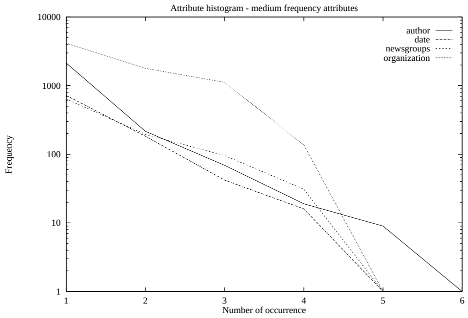

Figure 6: Attribute Frequency Histogram for Medium Frequency Attributes

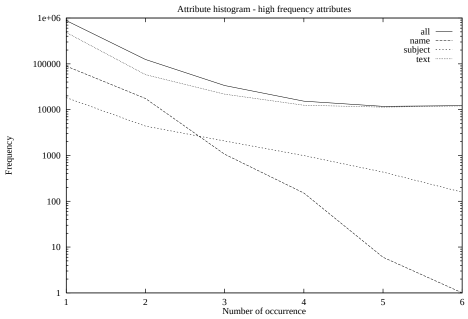

Figure 7: Attribute Frequency Histogram for High Frequency Attributes

ing and content routing are based on content labels, which are at present administratively determined.

For our initial prototype, we have chosen a simple routing algorithm based on partitioning the user's query into a *universe specification* and an *object specification*. The content router uses the universe specification to determine a set of information servers (some of these servers may be other content routers). It then forwards the object specification portion of the query to the above set of nodes. The nodes act on this query based on their function as either routers or end-point information servers. This type of naming is useful for users who want to browse the graph of information servers and/or refine a query so that a fewer number of servers need to be contacted. A complete query composed of the universe specification and the object specification will invoke the object query portion only on the set of information servers that match the universe specification.

The layered architecture is essential for scaling. As demonstrated in the previous section, it is impractical for content routers to maintain complete knowledge of the attribute information at the information servers. At the other extreme, maintaining no knowledge of the actual content of the information servers will result in broadcasting to all end-point servers in the attempt to process a query. Content labels represent a compromise between these two restrictions for a scalable architecture.

4.1 Content routing system implementation

Our content routing system uses the Semantic File System interface [GJSO91] for both information servers and content routers. Thus a content router is implemented as a user level NFS server that may be mounted by any NFS client. Path names are interpreted as queries, and results are returned in dynamically created *virtual directories.*

Figure 8 shows the architecture of our prototype content router implementation. the server collects together content labels from a set of servers to which it will provide access. Server transducing is currently done manually. Clients submit queries that pass through the NFS interface to the server. In response to user requests, the server constructs and returns virtual directories. On demand, the server computes the contents of virtual directories. In response to a universe specification, a query on content labels, the server provides users with access to the content labels that satisfy the query. When the user decides that a result set is sufficiently narrow, the server will forward queries to specified servers. It then returns the merged results in a virtual directory. To provide access to individual data objects and servers, the content router returns symbolic links which are interpreted by the automount daemon at the client. The client then uses its own mount point that directly connects to the remote information server. The client may also access objects through the content router if it does not run the appropriate automount code. Naturally, mediated object access comes at a performance penalty to be discussed below.

Our use of an NFS-based protocol has advantages and disadvantages. Because NFS is widely understood, and because our servers use NFS at both ends, we are able to reconfigure our system and use new machines easily. NFS also makes it very easy for clients to participate, and allows us to leverage existing code. One can run editors, compilers, and standard file system tools, all without modification. Unfortunately, NFS systems do not tolerate failure well as a rule, and error reporting and recovery are awkward in the protocol. Our architecture, however, is independent of the underlying communication protocol. We could use Gopher [AAL+91], World-Wide Web [BLCGP92], Z39.50 [NIS91], or any other mechanism that provides distributed access to servers.

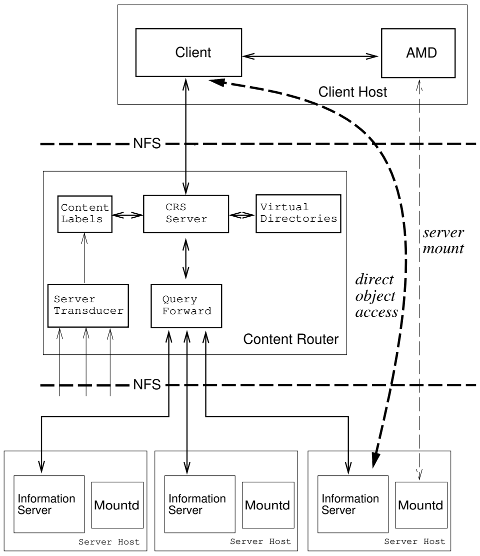

Figure 8: Implementation architecture for our prototype content router.

While a hierarchical system facilitates locating objects, it introduces communication delays and performance restrictions that become intolerable for larger sets of applications as the layering gets deeper. In particular, server load is proportional to the resources consumed by clients accessing the server. It is important for the performance of the system, and thus of client applications, to minimize the resources consumed by clients. Therefore, once a client identifies an object, or once the content router has discovered that there is only one information server appropriate to a client's request, the system must provide direct access to the end-point server. The client then does not incur the latency of a multilayered communications path, and the system does not have to provide resources that are not necessary to serving the client's request.

4.2 Performance results

We performed a suite of experiments with our server to test the performance implications of mediated versus direct access to objects. Figure 9 illustrates the performance penalty incurred when content routers mediate file read requests in our prototype. The figure plots throughput achieved at the client in Kbytes per second versus load at the content router. For our experiment, the load was generated by running 0, 1, or 2 processes that continuously generated lookup and read requests through the content router. Queries always go through the intermediate content routers. In one set of trials, *all* client requests were mediated by the content router, including file reads. For this set, the throughput for the case where the content router is not used is presented for comparison (above the label "direct"). In the other set of trials, the content router provided the client direct access to the server for object access.

The content routers provide unmediated access to data by returning symbolic links. These links are interpreted by a version of the Berkeley automount daemon [PW91]. Mediated accesses naturally go through the content router. In Figure 9, each data point represents the average of four trials each of which reads a 35 MByte digital video data file. The information server was running on an SGI 4D/320S. The content routers and clients were run on Sparc Station IPX's. All these machines were interconnected with a 10Mbit/s Ethernet. These figures compare with an average throughput of 830 KByte/s for standard NFS file access.

Two important observations can be derived from the data. First, as expected, the non-mediating servers do not hamper system throughput. Second, mediating routers degrade performance drastically, especially as the load at the content router increases. Mediated access has such a dramatic impact even for low loads because our implementation is not multithreaded and does not do read-ahead. That is, it is not optimized for providing file service. Nonetheless, the impact of content router load is inherent in any implementation that forces clients to use mediated data access. The more layers there are in the network, the more resources are consumed by each client, and the greater the likelihood that a client will suffer query router induced performance degradation.

The performance of query processing is acceptable. Since the number of servers supported at a content router is not very large (we expect on the order of hundreds), queries that return sets of servers can perform quite well. The performance of some sample routed queries is given in Table 2.

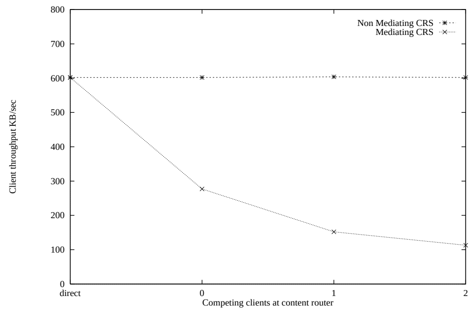

Figure 9: System throughput for mediating and non-mediating content routing system configurations

Q	$\#M$	#S	S_1	S_2	S_3	S_4	T_1	T_2
1	88	1	88				1.2s	0.4s
2	22	1	22				0.6s	0.2s
3	9	1	9				1.2s	0.2s
4	60	2	5	60			2.6s	0.8s
5	31	2	31	0			1.4s	0.4s
6	27	4	7	1	4	15	9.9s	4.5s

Table 2: Routed query performance

These performance measures also include an information server running on a heavily loaded Microvax 3500 (query 6), as well as the above mentioned machines. Q identifies a particular query, $\#M$ is the number of files that matched the query, $\#S$ is the number of servers the query was routed to, S_n is the number of results returned from server n, T_1 is the time it took to list the result set the first time (this includes the time to route and compute the query result set), and T_2 is the time it took to list the result set the second time (since the server caches the result set, $T_1 - T_2$ is the time to route the queries to the servers and compute the result set). Note that our current implementation does not route the queries in parallel.

From our experience thus far, we believe that query routing in a hierarchical collection of distributed servers is feasible. Even demanding applications such as ones using digital video can achieve adequate throughput provided content routers do not mediate object accesses.

5 Conclusion

We have described how data objects in distributed information systems can be located and accessed in an associative manner. To be efficient and scalable, associative access to distributed information systems must involve only a restricted set of relevant servers. The query search space even for simple queries can be enormous and thus can produce large result sets incurring high processing cost. For this reason, we have organized the search space using content labels describing servers and using attributes that have discriminatory power over servers. Content label based query routing is particularly suitable for navigating through a large space of servers. The visible structure and logical organization of the space implied by content labels can be discovered, and the scope of an initial query can be sufficiently narrowed and refined so that the result set is efficiently computed. Predicates provide widely varying assistance with formulation of user queries as an iterative process.

We have discussed how a layered architecture of content routers can be integrated into existing distributed systems. The layered approach is useful for scaling and administrative purposes as well as for organizing the global object search space into small manageable units that are easier to browse. Moreover, the architecture preserves the autonomy of individual servers: they propagate some knowledge about the content they provide, and they do not need to subordinate to some global organization scheme.

We have built a prototype based on the semantic file system with NFS as the underlying distributed system protocol. We have successfully used our system to locate data objects in several servers in our laboratory. Along with demonstrating the feasibility of our approach, the prototype shows good performance and allows users to enjoy the benefit of bypassing layers once objects of interest are located. The results of these experiments suggest that the content routing enables transparent search of distributed information without compromising access performance.

The results to date are consistent with our thesis that query routing based on content labels is an effective way to locate and access data objects in a distributed information system. We plan to enhance our prototype by adding automatic content label generation from selected attributes. Further analysis of statistics gathered from information servers will be used for choosing highly discriminatory attribute values for content routing. We also plan to examine methods for the exclusion of inaccessible servers which are inevitably present in any large scale distributed system. These servers should be treated in a special way because they may contain some currently inaccessible objects of interest. Another area of future research is exploration of how associative access can be used to locate replicated objects.

Acknowledgements

The authors would like to thank Brian Reistad and Franklyn Turbak for their comments on drafts of this paper.

References

[AAL+91] Bob Alberti, Farhad Anklesaria, Paul Linkner, Mark McCahill, and Daniel 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.

[BC92] Daniel Barbara and Chris Clifton. Information brokers: Sharing knowledge in a heterogeneous distributed system. Technical Report MITL-TR-31-92, Matsushita Information Technology Laboratory, Princeton, NJ, October 1992.

[BLCGP92] Tim Berners-Lee, Robert Cailliau, Jean-François Groff, and Bernd Pollermann. World-wide web: The information universe. *Electronic Networking*, 2(1):52–58, 1992.

[Cat92] Vincent Cate. Alex — a global filesystem. Usenix Workshop on File Systems, 1992.

[CCI88] CCITT. The Directory - Overview of Concepts, Models and Services. Recommendation X.500, 1988.

[DANO91] Peter B. Danzig, Jongsuk Ahn, John Noll, and Katia Obraczka. Distributed indexing: A scalable mechanism for distributed information retrieval. Technical Report USC-TR 91-06, University of Southern California, Computer Science Department, 1991.

[ED92] 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.

[GJSO91] David K. Gifford, Pierre Jouvelot, Mark A. Sheldon, and James W. O'Toole. Semantic file systems. In *Thirteenth ACM Symposium on Operating Systems Principles*. ACM, October 1991. Available as *Operating Systems Review* Volume 25, Number 5.

[KM91] Brewster Kahle and Art Medlar. An information system for corporate users: Wide area information servers. Technical Report TMC-199, Thinking Machines, Inc., April 1991. Version 3.

[Mar83] Richard 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.

[Mar90] Richard S. Marcus. Advanced retrieval assistance for the DGIS gateway. Technical Report LIDS R-1958, MIT Laboratory for Information and Decision Systems, March 1990.

[NIS91] Ansi z39.50 version 2. National Information Standards Organization, Bethesda, Maryland, January 1991. Second Draft.

[Pet88] Larry Peterson. The Profile Naming Service. *ACM Transactions on Computer Systems*, 6(4):341–364, November 1988.

[PW91] Jan-Simon Pendry and Nick Williams. Amd: The 4.4 BSD automounter reference manual, March 1991. Documentation for software revision 5.3 Alpha.

[SA89] Patricia Simpson and Rafael Alonso. Querying a network of autonomous databases. Technical Report CS-TR-202-89, Princeton University, Princeton, NJ, January 1989.

[Sch89] Michael F. Schwartz. The Networked Resource Discovery Project. In *Proceedings of the IFIP XI World Congress*, pages 827–832. IFIP, August 1989.

[Ste91] Richard Marlon Stein. Browsing through terabytes: Wide-area information servers open a new frontier in personal and corporate information services. *Byte*, pages 157–164, May 1991.

[WCS93] David C.M. Wood, Sean S. Coleman, and Michael F. Schwartz. Fremont: A system for discovering network characteristics and problems. In *USENIX Association 1993 Winter Conference Proceedings*, pages 335–348, 1993.