

Making Global Digital Libraries Work

Preface

This chapter is based on:

Lagoze, C., Fielding, D. and Payette, S., Making Global Digital Libraries Work: Collection Service, Connectivity Regions, and Collection Views, in *ACM Digital Libraries '98*, (Pittsburgh, 1998) [291].

This chapter describes work on the Distributed Interactive Networked System For Technical Reports (Dienst), which began in 1992 under the auspices of the DARPA-funded Computer Science Technical Reports Project (CSTR)⁶⁰. This project was a collaboration between the “five leading US computer science departments” – Berkeley, Cornell, Carnegie Mellon, MIT, and Stanford – and was administered by the Corporation for Network Research Initiatives (CNRI). The goal of this early digital library project was to develop the technology and understand the intellectual property issues for putting computer science technical reports online.

As described in Chapter 5 the results of the work had a substantial impact on later digital library architectures and implementations. Dienst was at one point proposed as the basis for interoperability among all projects funded in the Digital Libraries Initiative [330]. In addition, it was widely deployed worldwide in the Networked Computer Science Technical Research Library (NCSTRL) [153], which at its height included over 160 institutions. The work also led to a collaboration with the ePrint

⁶⁰ <http://www.cnri.reston.va.us/cstr.html>

arXiv⁶¹, then at Los Alamos now at Cornell, to develop the Computing Research Repository (CORR) and incorporate that into NCSTRL [226].

The Dienst architecture and protocol was notable for a variety of reasons. The development of Dienst occurred shortly after the introduction of the Mosaic browser, commonly acknowledged as a main factor that caused the initial popularization of the web. Before the existence of web technologies and the notion of a common browser client, client/server systems had to confront the difficult task of creating cross-platform clients and architecting client/server interactions at a lower level in the Internet stack (e.g., at the socket level). Dienst was the first digital library architecture to fully leverage these newly introduced web technologies, including URLs, HTML, HTTP, and the increasingly ubiquitous web browser.

In addition to this innovative use of web technology, Dienst incorporated a number of interesting concepts that had an important effect on later digital library developments. These include a digital object model that included the notion of compound, multiple-data stream resources and protocol-based accessibility to this compound objects; simple bibliographic metadata as an alternative to heavyweight library cataloging records; federated search, which is a special focus of this particular paper; and an extensible distributed service model. All of these technical developments are described in greater detail in the body of this chapter.

The chapter, based on a 1998 paper, reveals a number of the assumptions about digital libraries and the web, which were common in that early time, as described in the first part of this dissertation. It emphatically makes the point of distinguishing the web from a digital library, stating “a distinguishing aspect of a library (digital or otherwise)

⁶¹ <http://arxiv.org>

is management of collections. Given this understanding of a library the World Wide web is NOT a digital library. It represents a set of objects joined together technically (by the common protocol HTTP), but not by any collection management actions.”

Dienst can be characterized as a “classic” digital library architecture, revealing many artifacts of the traditional library meme. Search is based on catalog records, formatted with a simple metadata vocabulary [322], a precursor to the later Dublin Core vocabulary [482]. The entry point to search is through a portal, a “user interface server”, and mechanisms to expose the collection to (then primitive) web search technologies is not considered. Digital objects are organized into collections, and stored in and accessed from distributed repositories.

The focus of this paper, federated or distributed indexing and searching, reflects a particular artifact of the time and mindset of the digital library community, and is particularly demonstrative of these early assumptions, which now seem ill founded.

The following quote from the paper represents the core of these assumptions:

In the present World Wide Web, virtually all tools for resource discovery are based on a centralized model. Typically, a central service creates and deploys a master index, and sometimes creates one or more replicas of the index. Although this model is currently prevalent, we argue that distributed searching will become increasingly necessary to overcome the constraints inherent in the centralized model.

The paper goes on to justify this statement on the grounds of scalability, customizability, and intellectual property. Written before the explosive growth of Google and other crawler-based search indexing, these statements clearly failed to anticipate the complications of distributed searching both in terms of rank merging and reliability and the real scalability of centralized indexing. We described some of the performance issues with distributed searching in other papers [170, 171].

Another relevant concept touched on in the paper is the problematic and ambiguous notion of the control zone [34] in the digital library. Quoting from the body of the paper:

... in the traditional library model, some librarians argue that physical containment of objects (e.g., in stacks) is the primary criterion for inclusion in the collection. This notion of physical control breaks down in the networked environment of digital libraries where both overt and implicit linkages can be made between objects that reside in different physical locations.”

Note that the response to this problem in Dienst was to define the control zone in terms of the digital library catalog: “An object is ‘in’ a digital library's collection if it can be directly discovered using the resource discovery tools defined and implemented by the respective digital library”. Given the present reality where virtually all resource discovery takes place in mainstream search engines, rather than digital library catalogs, this definition certainly seems anachronistic.

Acknowledgments

The work described in this chapter was funded by the Defense Advanced Research Project Agency under Grant No. MDA 972-96-1-006 with the Corporation for National Research Initiatives. The Dienst architecture was originally formulated in 1992 by James R. Davis, then at the Xerox Design Research Institute at Cornell. He continued to make substantial contributions to this work through 2000. Other contributors to this work, all members of the Cornell Digital Library Research Group, were David Fielding, who collaborated on the design of the collection service and wrote the initial implementation of it, Naomi Dushay, and Sandy Payette, both of whom participated in the collection service design and worked on other parts of the Dienst architecture. Acknowledgments also go to Dean Krafft whose thoughtful observations inspired a good deal of the work at the Cornell Digital Library Research

Group. Finally, thanks to Bill arms at CNRI for his helpful feedback and support on this work.

Introduction

Since 1993 the Cornell Digital Library Research Group has been investigating the architecture of globally-distributed, federated digital libraries. In contrast to centralized or replicated stand-alone systems, these federated systems are composed of semi-autonomous services, distributed across the global Internet, that interoperate through an open protocol.

From the point of view of flexibility, extensibility, and scalability this federated model is preferable to self-contained, centralized systems (such as the current generation of library management systems that form the technical basis of modern libraries). Among the benefits of the federated model are:

- Stakeholders can maintain control of digital objects (documents) in their own repositories.
- Customized collections can be created by aggregating digital objects in these distributed repositories.
- New value-added services can be created as the need arises.
- The functionality of existing services can be enhanced in a modular fashion.
- Services can be replicated to enhance global accessibility.
- Customized user interfaces (digital library gateways) can be created to provide community-tailored access to other distributed digital library services.

These advantages gained by modularity, interoperability, and distribution should not come at the cost of decreased usability or performance. As much as possible, users should be insulated from the physical distribution of the system and should be able to

view the digital library as a single collection with uniform tools for search, retrieval, and display of information within. At the same time, the performance of the system should match user expectations. This "illusion of uniformity" should be maintained, whenever possible, in the face of poor and inconsistent network connectivity, variability in server load, inconsistent server administration, and other problems characteristic of distributed, decentralized systems.

Maintaining usability in the presence of such distribution is one of the key challenges for designing digital library architecture. Some of the aspects of this challenge have been extensively covered in the distributed systems literature [59]. However, issues of global scale and a high degree of component autonomy change the flavor of the digital library problem sufficiently to call for some new solutions.

In this paper we examine one aspect of the distributed digital library problem - distributed searching. In the present World Wide Web, virtually all tools for resource discovery are based on a centralized model. Typically, a central service creates and deploys a master index, and sometimes creates one or more replicas of the index. Although this model is currently prevalent, we argue that distributed searching will become increasingly necessary to overcome the constraints inherent in the centralized model. In particular, effective architectures for distributed searching must be developed to address:

- *Issues of Scalability* - As the global information space explodes, it has become increasingly difficult to collect indexing information and keep centralized indexes up-to-date. Commercial web search providers are beginning to recognize this fact and it has even lead to a recent commercial patent for distributed searching technology [244].

- *Issues of Specificity* - While interoperability among search or indexing sites is important, it is also vital that the information infrastructure accommodates the unique needs of specific communities. Such accommodation is best accomplished via separate service providers that can both cater to individual community needs (through custom metadata, specialized data formats, query languages, user interfaces, etc.) and interoperate on a global scale through open protocols.
- *Issues of Intellectual Property* - The current resource discovery infrastructure depends on the fact that almost all the items in the global information space are not encumbered by access restrictions. Certainly this will change as improved technology for digital object rights management evolves [30, 446] and, as a result, more objects with more restricted access proliferate on the net. (In fact, one could argue that in the future the objects with the most value will be those that are not freely available.) In this case, it will become more difficult if not impossible for centralized search providers to collect indexing information by simply walking the global information space (in the fashion of current "web spiders"). As a result, resource discovery will depend on distributed indexing sites that are physically, logically, or legally linked (through licensing agreements) with sites of content providers.

Other researchers have investigated a variety of issues relevant to distributed searching. The distributed database community has a long history of investigating the optimal distribution of indexing information across LANs and controlled WANs [125]. Researchers in the digital library community have examined query translation issues [116], content summarization for query routing [216], and protocols for meta-searching and metadata collection [215].

This paper describes an architecture, and experience with that architecture, for distributing index servers⁶² on a global scale and disseminating meta-information on the location of those servers among participating servers. The architecture has three logical components. The first is a distributed collection service that identifies the index servers of a distributed digital library collection and manages meta-information about those servers. The second is a connectivity region, which is a set of nodes on the Internet with relatively good network connectivity (e.g., low latencies, infrequent partitioning). The last is a collection view, which is a perspective on a collection specific to a connectivity region.

The architecture described here was developed out of our experiences at Cornell building a globally distributed digital library, NCSTRL⁶³ (Networked Computer Science Technical Reports Library). The structure of this paper reflects the development path of the Cornell work. First, we briefly summarize NCSTRL, our global test bed, and Dienst, the technology on which that test bed is based. This section includes a description of the initial Dienst collection service, which forms the basis for the expanded collection service described later in the paper. We then describe our early efforts, or mistakes (depending on your perspective), to deal with distributed resource discovery in NCSTRL. Following this we describe the current evolution of our distributed searching architecture, with an explanation of connectivity regions and how they are implemented using an enhanced collection service. We conclude by describing some future work and opportunities for research.

⁶² Throughout this paper an *index server* is a server that collects meta-information about objects in a digital library collection and returns results (hit lists) in response to queries on that meta-information.

⁶³ Pronounced "ancestral".

NCSTRL – The test bed for a globally distributed digital library

The global digital library architecture described in this paper is the result of our work with Dienst [152], a protocol and a reference implementation for distributed digital object libraries. The initial Dienst system was designed and developed as part of the DARPA-sponsored CS-TR project⁶⁴, which investigated general digital library issues and, in particular, the technology for making technical reports digitally available from the participating institutions⁶⁵.

At the conclusion of CS-TR funding, participants in WATERS [356], one of several other efforts to create a digital library of computer science technical reports, joined with developers of Dienst and other members of CS-TR to form NCSTRL. By June 1998, NCSTRL had grown to include collections from over 100 institutions with over 60 servers worldwide. The globally distributed nature of NCSTRL and the federated, open architecture of Dienst on which it is based, represents a unique test bed for ongoing digital library experiments. Those experiments are both of a technical nature, such as those described in this paper, and of a social nature, exploring the organizational aspects of loosely federated information systems.

Dienst architecture

The remainder of this section summarizes the Dienst architecture that underlies NCSTRL. A more detailed description of Dienst can be found in the Implementation Reference Manual [302]. The fundamental features of the architecture are a logical document model, distributed digital library services, and an open protocol for interoperation among those services.

⁶⁴ <http://www.cnri.reston.va.us/cstr.html>

⁶⁵ U.C. Berkeley, Carnegie-Mellon, Cornell, M.I.T., and Stanford.

Logical Document Model

At the core of the Dienst architecture is the notion of a document, a logical abstraction⁶⁶ that incorporates a number of concepts.

- Each Document has a *globally unique name* that is defined using the handle service⁶⁷.
- A document consists of a number of *components*. The two components currently in use in NCSTRL are the bibliographic description and the "body" of the document.
- Each component is available in one or more *formats*. For example, the body of the document may be available in PostScript, HTML, and as a sequence of TIFF images.
- A component in a format may be divided into a number of *decompositions*. For example, the "body" available in PostScript format may be divided into "pages" or "chapters".

Digital Library Services

The functionality of the Dienst architecture is logically divided among a set of distinct services. Although the services are modular in nature, they are currently implemented as a single physical server. This grouping of services was merely a matter of expediency, and our current research and development efforts are motivated by our belief that the digital library service structure should be physically, as well as logically, modular.

⁶⁶ By *logical* we mean that the abstraction is distinct from the "physical" one-to-one mapping of document to file that exists in file systems (local or distributed), FTP, or HTTP (sans CGI).

⁶⁷ <http://www.handle.net/>

There are three core Dienst services, in addition to the collection management service that we describe in the next section. The core services are:

- The *Repository Service* that stores and provides access to documents identified using the global naming service and structured according to the document model described earlier,
- The *Index Service* that stores indexing (meta) information about documents in the collection and responds to queries on this indexed information, and
- The *User Interface Service* that provides a human front-end to the other services.

Open Protocol.

Dienst services and servers interoperate using a well-defined protocol [151]. The structure of this protocol corresponds to the logical services described above. Each protocol request is framed as a verb to a service.

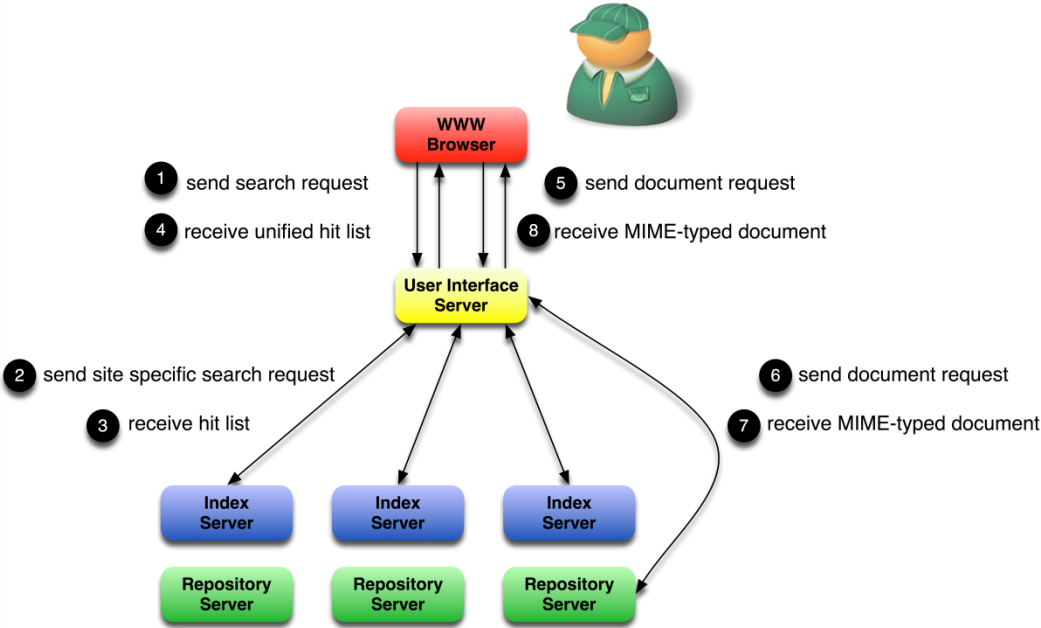


Figure 24 - Dienst Services

Figure 24 illustrates the protocol-based interactions between the core Dienst services for search and retrieval of a document. The user interface service acts as the mediator between the user's browser and the multiple index and repository services in a distributed collection. Requests are made through verbs addressed to one of the three core services. For example:

- The *Search* verb of the index service returns a citation list of documents meeting specified search criteria in the respective index
- The *Fetch* verb of the repository service returns a *dissemination* of a specified document identified by its unique identifier (or handle). Arguments to the *Fetch* verb conform to the concepts in the Dienst document model. For example, a *Fetch* request may specify that page one of the document body should be returned in GIF format.

The use of an open protocol has two key advantages. First, it permits the construction of other value-added services that interact with existing Dienst servers. Second, it allows individual services and, in fact, the entire Dienst implementation to be replaced as other alternative implementations are developed.

The open protocol is a defining feature of the collection service we describe next. This service is critical to the management of a distributed digital library collection in the Dienst architecture.

The Collection Service: Defining the Contents of the Digital Library

A distinguishing aspect of a library (digital or otherwise) is management of collections. Management of the collection begins with selection of the objects to be included in the collection. Objects are selected from a global information space (e.g., the set of all published books, or the set of all objects on the Internet), and become constituents of library collections based on criteria applied by selectors or collection

managers. Depending on the sophistication of the library, there may be other collection management functions such as preservation, archiving, and the like. Given this understanding of a library, the World Wide Web, by itself, is NOT a digital library. It represents a set of objects joined together technically (by the common protocol HTTP), but not by any collection management actions. Similarly, sets of documents residing on servers communicating via the Dienst protocol do not comprise a digital library.

Thus, digital libraries cannot be defined by the mere existence or application of enabling technologies. Digital libraries are distinguished from the more ubiquitous networked information landscape through their incorporation of collection management services, which may involve human intervention.

Even with collection management, the definition of what is actually “contained” in a digital library can become ambiguous. For instance, in the traditional library model, some librarians argue that physical containment of objects (e.g., in stacks) is the primary criterion for inclusion in the collection. This notion of physical control breaks down in the networked environment of digital libraries where both overt and implicit linkages can be made between objects that reside in different physical locations. For example, if object A is included in a collection, are objects B, C, and D that are linked to object A also included in the collection? If so, are all objects transitively linked to object A via other objects also included? The answer to these questions has important implications in the areas such as legal responsibility and public service.

While there are, undoubtedly, multiple perspectives on the definition of digital library collections, in this paper we will adopt the following working definition. An object is

"in" a digital library's collection if it can be directly discovered using the resource discovery tools defined and implemented by the respective digital library⁶⁸.

We emphasize the distinction between discovery and retrieval in this definition. First, discovery of the object may mean that a surrogate of the actual object may be indexed, and the actual object (which may be a physical artifact) must be retrieved through other means. Second, assuming a global name space, any object in the global information space may be retrievable (using its URN) without necessarily being in the library from which it is being fetched. One can think of this type of retrieval as a type of digital "inter-library loan".

Another interesting aspect of collection building is the level at which "inclusion" is evaluated. At the lowest level, individual digital objects are aggregated to form a (sub)-collection. At a higher level, multiple (sub)-collections of items are federated to form larger collections.

NCSTRL is a working example of multiple levels of collection management. The Dienst architecture provides for institutional autonomy in item-level collection building, and the capability for institutions to federate into the larger NCSTRL collection. The Dienst collection service is the mechanism for managing the federation level of collection definition. The data for managing the collection is obtained via protocol requests to this service that return the following information:

- *The list of organizations that are part of the collection.* In NCSTRL the granularity of an organization corresponds to the computer science departments and research institutions that are members of NCSTRL (*e.g.*,

⁶⁸ This brings up the interesting question whether the set of objects discoverable through one of the web search services is part of the digital library defined by that service. The nature of digital libraries and their collections provokes many interesting questions.

Cornell Computer Science Department, Georgia Institute of Technology
College of Computing).

- *The network location.* The service provides the address and port of the Dienst index servers that store indexing information for each organization. For example, indexing information for Cornell Computer Science may be stored at foo.ncstrl.org port 80 and bar.ncstrl.org port 8083.
- *Meta-information about each of the index servers.* At present this meta-information indicates whether the index server should be considered primary or secondary. However, our intention is to expand this meta-information to include data about last update of the index, performance information, content summaries, and the like.

From the administrative perspective, the collection service allows easy management of the NCSTRL collection. Organizations join NCSTRL by submitting an application to our collection librarian⁶⁹ via the Web. Following our confirmation that the organization conforms to the collection profile (the institution should be a Ph.D. granting institution in computer science) and has a working Dienst-protocol-conformant server, the NCSTRL administrator at Cornell adds the institutional information to the collection service tables. This new institution then becomes visible to each NCSTRL user interface server after its next collection service request.

We originally implemented the collection service on a single Dienst server. In this configuration, the address and port number of the collection server is stored in the configuration file of each Dienst server. Periodically (every hour) each Dienst server issues a collection service protocol request to obtain the collection information, which

⁶⁹ Rebecca Wesley at Stanford University.

we described earlier. The requesting Dienst server then stores the collection information internally in a table. At this point the user interface services have access to the current list of participating organizations (provided by the collection server). Figure 25 illustrates the interaction between the collection service, user interface servers, and index servers in Dienst. As shown, each user interface server queries the collection server for collection information. For a specific query, an individual user interface (labeled UI₁ in the figure) uses this collection information to determine which index servers should process the query.

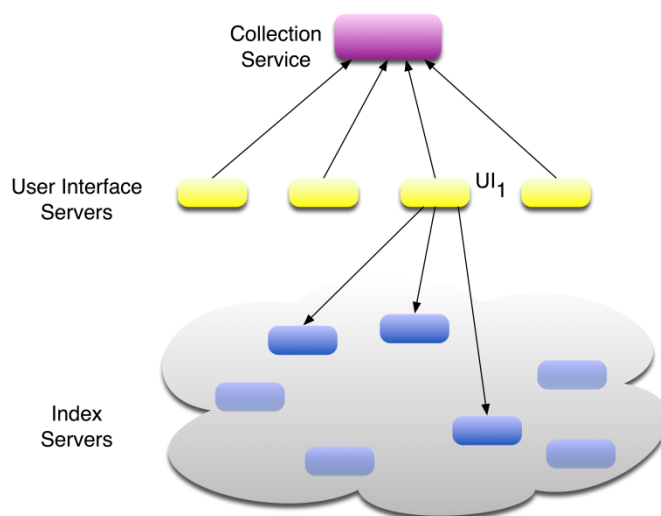


Figure 25 - Dienst service interactions.

From a user perspective, the latest organizations appear on the search form provided by the user interface service. When composing queries to the NCSTRL collection, users choose which organizations should be included in the search results. The respective user interface service can determine where queries should be dispatched by using the network location and contents data provided by the collection service. Once this information is obtained, the user interface service submits the actual query to the target index servers using a Dienst index server protocol request. When responses are

returned from the target index servers, the user interface service merges the responses into a single result set.

The next section of this paper describes the use of the collection server to implement two initial distributed search topologies in NCSTRL. The section that follows then describes a distributed version of the collection service based on connectivity regions, enabling globally distributed search.

The evolution of a distributed digital library: early experience

The flexibility of the collection service and its interaction with the user interface services allowed us to rapidly expand the NCSTRL collection from five sites in 1995 to over 100 sites in 1998. In the course of this rapid expansion, we implemented two initial distributed searching topologies: simple distributed searching and distributed searching with backup. In this section we briefly describe those topologies, and the lessons learned from deploying them in NCSTRL.

Some of the architectural solutions described in this section may, in hindsight, seem rather naïve and the results predictable. While that may be true, these solutions were developed and retrofitted onto a rapidly growing production distributed system. In addition, the experience gained from this incremental approach proved valuable and helped contribute to the architectural solutions described later in this paper. Finally, some have argued that given the present scale of the NCSTRL collection, centralized replicated searching is the more preferable and predictable model⁷⁰. This may also be true. However, as we argued earlier in this paper, the distributed searching problem will have to be investigated for future digital library infrastructure to operate, and NCSTRL has been, and still is, a unique test bed for researching those issues.

⁷⁰ Ed Fox, personal communication.

Simple Distributed Searching

The five institutions that participated in the CS-TR project and that participated in the initial Dienst-based collection shared two characteristics having implications for distributed searching reliability:

1. **Connectivity.** Among the five institutions connectivity was good; network down-time was minimal and latencies were fairly low.
2. **Commitment.** Due to joint funding within the CS-TR project, these five institutions shared a common interest and commitment to the success of the test bed technical report collection. As a result, the five servers and their contained collections were well administered.⁷¹

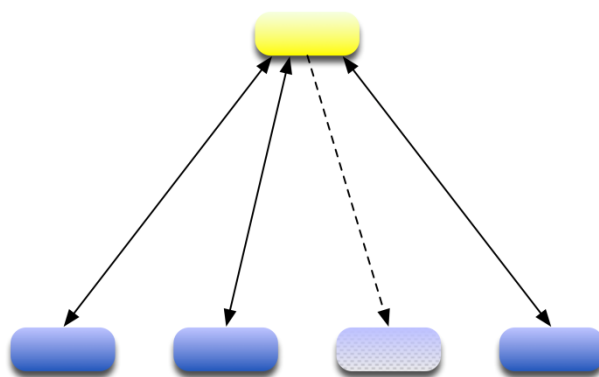


Figure 26 - Simple distributed search with server failure

Based on the high technical and administrative reliability, we made the initial decision to implement a simple distributed searching topology. In this topology only one index server existed for each organization in the collection. In fact, the index server was

⁷¹ We strongly emphasize that the factors that contribute to the success of a federated library are not restricted to the technical domain. We have found throughout the existence of NCSTRL that poor management of a few individual servers in a federated system can seriously degrade the reliability and integrity of the entire system. Poor management can take a variety of forms including a server that is periodically unavailable, descriptive metadata that is incomplete or incorrect, a collection that is not kept up to date, or any number of other factors.

resident in the same Dienst server as the document repository that it indexed. A search query from any of the user interface servers in the collection was, regardless of the origin of the search, dispatched to the same set of indexing servers. If an individual indexing server was unavailable (due to network failure or server failure) or overloaded (resulting in a time-out) the user was alerted that results could not be returned for the organization stored on that index server.

This simple topology is illustrated in Figure 26, with a connection failure to one of the index servers. The loss of access to information resulting from unavailable or slowly responding servers motivated the introduction of backup servers into the distributed searching scenario.

Distributed Searching with Backup

Even with a controlled set of servers, as was the case in the original CS-TR project, server failures occurred too often. As the size of the collection grew beyond the original five institutions, the number of failures increased dramatically. In fact, most search result sets were incomplete, showing one or more "unavailable organizations".

In response to this situation, we soon introduced replicated index servers, with a ranking of which server was primary, secondary, etc. This was done by extending the collection service protocol so that it could indicate the priority order of a specific index server for a specific organization. For example, the protocol response might indicate that `foo.ncstrl.org` port 80 is the primary index server for the Cornell CS collection, but `bar.ncstrl.org` port 8083 is the secondary index server for that same collection. Using this information, an individual user interface server could then first distribute the search request to the appropriate set of primary index servers. In case of failure or time-outs, the user interface could then distribute the same to the secondary

index servers corresponding to the "unavailable organizations" in the primary phase of the search.

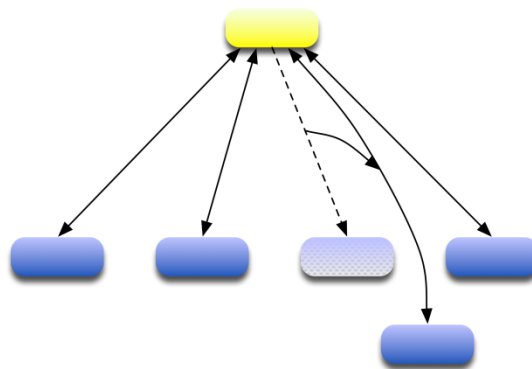


Figure 27 - Primary and secondary index servers

Figure 27 shows an example of this primary and secondary (backup) index server topology. In the illustration, one of the index servers has failed, and the query is redirected to the secondary index for that site.

Adaptive Routing between Primary and Secondary Index Servers

Experience with the backup index server topology demonstrated that in many cases poor performance or failure of an individual server persists over time. For example, a network or server failure is normally not repaired immediately. Rather than continuing to use a failing primary index server, it is preferable that the user interface server "remember" the failure of the respective server and, as a result, change the rank ordering of the index servers.

To implement such behavior we implemented a simple adaptive algorithm at each user interface server that keeps track of the success or failure history of each index server to which a queries are routed. If a specific index server repeatedly fails within a specified period and a secondary index server exists for the organizations indexed by that server, the unreliable server is "demoted" and the appropriate backup index

servers "promoted". This change in rank is left in place for a fixed period, after which the demotion and promotion are undone (but re-instated if the next retry results in another failure). In this manner, the overall response time to queries is relatively insulated from the effects of unreliable servers.

Connectivity regions and distributed collection service

The addition of international partners to NCSTRL, and the resulting global deployment of Dienst servers, required rethinking the ranked index server topology described in the previous section. As is well known, global connectivity varies dramatically. In fact, the latency times between nodes can differ by several orders of magnitude. In addition, the patterns of connectivity are not necessarily geographically related. Points that are coincident in physical space may be "distant" in network space, as measured by reliability and speed of the connection. This disparity between geographic and electronic "proximity" often corresponds to patterns of telecommunication development over the past fifty years, which often corresponded to political and colonial patterns. The exaggeration of this pattern is the fact the phone (and network) connections from a developing country to its former colonial power are in most cases better than to its neighbors (or, in fact, within its own country!).

We model the patterns of global connectivity through the notion of a connectivity region. A connectivity region is defined as a group of nodes on the network that among them have good connectivity⁷², relative to nodes outside of the region. At present, this definition is qualitative, but we plan to develop a more quantitative definition of the concept. The meaning and purpose of the connectivity abstraction is orthogonal to whether the region is statically or dynamically (adaptively) defined.

⁷² For the remainder of this paper we will define the quality of connectivity as a factor of both latency and reliability (resistance to failure).

The concept of connectivity regions allows us to reframe the requirements for distributed searching in the following fashion. In the absence of network or server failures, query routing from a specific user interface site should be restricted to those index servers in the same connectivity region. In case of a failure, an alternative indexing server should be chosen either in the same region or in another region with which there is good connectivity.

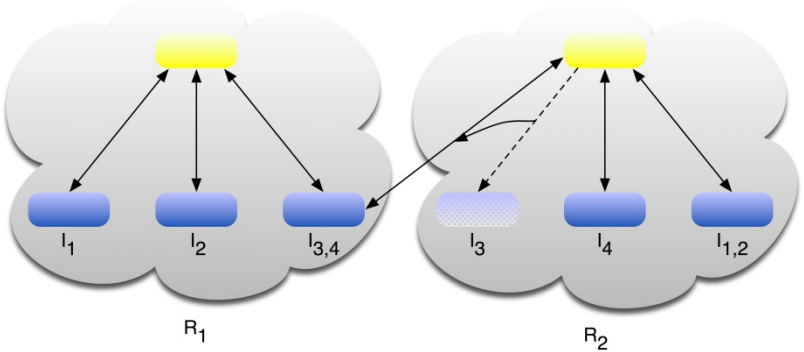


Figure 28 - Connectivity regions

Figure 28 illustrates a simple example of connectivity regions and the motivation behind them. In this figure there are two regions, labeled R₁ and R₂. Each region contains one user interface server, which dispatches queries and combines responses, and three index servers, which respond to queries. In the example, the indexed data in the collection is divided into four partitions, and the subscript(s) on each of the indexing servers indicates the partition(s) indexed at the index server. For example, index server I₁ holds indexing information in partition 1 and index server I_{3,4} holds indexing information in partitions 3 and 4. As illustrated, indexing information is replicated in a manner that queries can be routed within a region in which a user interface server is located. However, as also illustrated, a failure in an index server in a region (I₃ in R₂) may require routing of a query to an index server outside the region (I_{3,4} in R₁).

A Distributed Collection Service and Collection Views

Earlier in this paper, we described how Dienst user interface services use data from the collection service to determine where to route queries. The routing is both content based - which index servers can answer queries for the organization(s) specified in the query - and priority based - which index server(s) should be considered primary, secondary, etc. for the specific organization(s). As described, the original collection service was implemented within one server. All Dienst user interface servers in the collection used that single server as the source for collection data. Furthermore, the collection data supplied to each Dienst user interface server was identical.

In contrast, the connectivity region concept, illustrated in Figure 28, implies that the routing decisions made by different user interface servers may be based on different collection information. In the example, the user interface server in region R_1 "believes" that the primary source for indexing information on partition 1 of the collection is at the index server labeled I_1 . On the other hand, the user interface server in region R_2 "believes" that the primary source for that information is at the index server labeled $I_{1,2}$. In other words the collection view, the meta-information about the contents of the collection, of the R_1 user interface differs from that of the R_2 user interface. A single collection, such as NCSTRL, may have multiple collection views, corresponding to the connectivity regions that have been defined for the servers in that collection.

In order to support the notion of multiple collection views, we re-implemented the Dienst collection service in a distributed manner. In this new implementation, the distributed collection service was divided into two logical server types.

- 1) *Central Collection Server (CCS)*. There is a single central collection server that serves as the central point of management of the collection. This server stores the following information:
 - (1) A table defining all organizations in the collection, in the same manner as the original collection service implementation.
 - (2) The list of Dienst servers (identified by host and port) that are acting as regional collection servers. There is one regional collection server per connectivity region.
 - (3) A set of collection views, each one corresponding to a defined connectivity region. Each collection view contains the list of index servers that should be used (along with their rank orders) by the user interface servers in that region.
- 2) *Regional Collection Servers (RCS)*. As described above, there is one RCS per connectivity region. An RCS provides the same collection information to the user interface servers in its region as the original single-site collection service. That is, it returns to them the set of rank-ordered index servers that they should use for query routing. Like the original implementation, the RCS gets the information from the CCS, which returns the collection view that corresponds to that region.

Figure 29 illustrates the interactions between the CCS and RCS and Dienst user interface servers. As shown, the central collection server (labeled CCS) contains internal tables that store, for each connectivity region, the server address of the RCS for that region and the collection view that corresponds to that region. In the figure, the RCS labeled S_1 (which is configured with the CCS as its collection server) submits a protocol request to the CCS to fetch a collection view. The CCS, recognizing S_1 as the RCS for R_1 , returns the appropriate collection view. The user interface server in R_1 ,

which is configured with S_1 as its collection server, then receives the correct collection view in response to its collection service protocol request to S_1 . It then uses the information in this collection view to make routing decisions to index servers. It should be noted that as network connectivity changes, the regional view could be re-defined at the CCS level. A region's collection view is modified once the RCS requests and receives new collection data from the CCS.

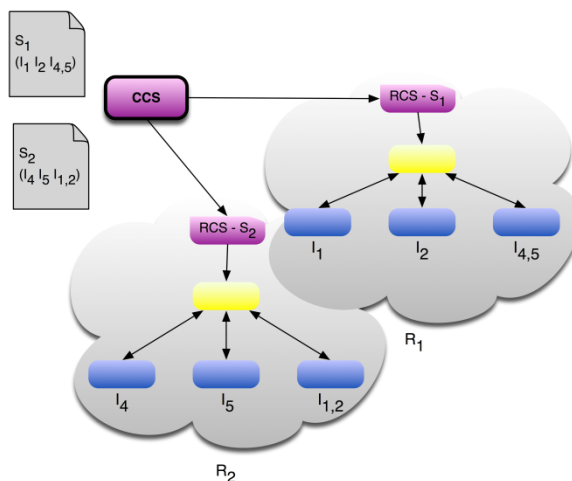


Figure 29 - Interactions of CCS, RCS, and user interface server

There is one final implementation note. If a server not listed with the CCS when an RCS submits a request to the CCS for a collection view, the CCS returns a view that is registered as the "default" collection view. In this fashion, any external service or agent can make use of the collection service for its own internal purposes.

Experience with the Architecture

At the time of completion of this paper in 1998, we had four operating regions within NCSTRL. A server operated by MTA/SZTAKI in Budapest, Hungary acts as an RCS for Dienst servers in Eastern Europe and Italy. A server operated by ICS/FORTH on Crete acts as an RCS for Dienst servers in Greece. A server operated by GMD in Bonn, Germany acts as an RCS for Dienst servers in Northern Europe. A server

operated by Cornell Computer Science in Ithaca, NY acts as an RCS for North American and some European servers with good Trans-Atlantic connections. We have found that connectivity between the West (especially the San Francisco area) and East coasts of the United States is often as bad as between Europe and North America. Because of this, we are investigating breaking up the North American region into two or possibly three regions.

The current configuration of regions was based mainly on conjecture, informal experience, and the willingness of particular Dienst sites to assume the greater reliability responsibilities required by an RCS. Thus, any conclusions based on our experiences are preliminary. In any case, we have found that the perceived reliability of the Dienst system, as measured from any of the user interface gateways, has improved dramatically. In addition, the architecture has proven quite easy to manage and adjust. Modifications to tables in the CCS are quickly propagated to the RCS's and thus to the Dienst servers in those regions. A server can easily be moved from one region to the next and the effect of unreliable servers can be isolated.

Our initial implementation of connectivity regions has uncovered a number of problems that we intend to address in future implementations.

- The implementation was retrofitted on top of a Dienst protocol and Dienst servers that pre-dated the regional architecture. Because of this we had to make a number of implementation compromises to avoid "breaking" legacy systems⁷³. One example of a problem that we have had is the imprecise and insufficient information about database freshness supplied by existing Dienst

⁷³ This is not an uncommon problem with distributed software for which a satisfactory solution will need to be found.

servers. This has made it difficult for us to propagate up-to-date replicas of indexing data between index sites.

- Connectivity problems remain troublesome. For example, the network speed between our Hungarian RCS and other Dienst systems is sometimes so bad that it is impossible to update index servers in that region.
- Server administration problems make it difficult to maintain index server integrity. When the primary source of indexing information is frequently unavailable or the quality of records is inconsistent, it is impossible to maintain useful replicas of that information.

As one strategy for eliminating these problems we are planning to logically segregate our production system from our research test bed. In this manner we can maintain production NCSTRL services, perhaps with a more centralized search strategy, and carry out research on isolated and controlled servers in the test bed. Ironically, the regional architecture can be used to create this segregation - in effect breaking off "production regions" from "research regions".

Finally, researchers outside of Cornell have experimented with the regional idea. For example, the MeDoc project has adapted the concept for defining content-specific regions or collections [16]. Researchers at ICS/FORTH in Greece and at IBM-Watson have used it in experimentation on QoS-based Searching and Retrieval [427].

Conclusions

As stated earlier in this paper future digital libraries architecture will have to address the problems inherent in distributed searching. They will have to do this in the context of global connectivity patterns. Our experience with NCSTRL has shown that the digital library infrastructure must provide information that supports query routing

decisions. Using this information, individual services can then algorithmically or heuristically decide the "best" destination(s) for protocol requests.

In the process of implementing and deploying Dienst and NCSTRL we have developed a number of useful abstractions for addressing this problem. This chapter has described three concepts that together have allowed us to globally distribute the NCSTRL collection.

- The *Collection Service* defines for user interface gateways the location of servers to which resource discovery queries can be routed.
- *Connectivity Regions* define the division of the complete set of servers into groups with relatively good connectivity characteristics.
- A *Collection View* is a definition of the collection, framed as the location of index servers, which corresponds to the connectivity characteristics of a connectivity region.