Dienst: Building a Production Technical Report Server

James R. Davis[*] Dean B. Krafft[†] Carl Lagoze[‡]

Abstract

Dienst is a protocol and implementation that provides Internet access to a distributed, decentralized multi-format document collection. It supports full text and boolean searches, thumbnail visual browsing, on- screen reading, and printing. In this paper we describe a number of issues that arose as we developed Dienst, including copyright policy, document management, and handling multiple formats of documents, and then lay out a number of areas where we still need to make progress, including interoperability among heterogeneous search engines, reliability, logical document structuring, and technology transfer.

1 Introduction

In this chapter, we describe our experiences with designing, implementing, and distributing Dienst. Dienst [2] is a protocol and implementation that provides Internet access to a distributed, decentralized multi-format document collection. The

[*]Xerox Corporation, Design Research Institute, Cornell University, Ithaca, NY 14853, davis@dri.cornell.edu

[†]Computer Science Department, Cornell University, Ithaca, NY 14853, dean@cs.cornell.edu

[‡]Computer Science Department, Cornell University, Ithaca, NY 14853, lagoze@cs.cornell.edu

collection is managed by a set of interoperating Dienst servers distributed over the Internet. These servers manage three basic digital library services: repositories of multi-format documents; indexes into the document collection and search engines for these indexes; and user interfaces to provide front-end services for browsing, searching, and accessing the collection.

In the sections below we give a brief overview of Dienst, we describe a number of areas where Dienst has made a significant contribution to the state-of-the-art, and we then discuss some areas of current research and development. We conclude with a look at our plans for the future of Dienst.

2 Overview of Dienst

The Dienst architecture was created by a team from the Cornell University Computer Science Department and Xerox Corporation as part of the Computer Science Technical Report (CSTR) project. This project is an ARPA-funded consortium consisting of the computer science departments at UC Berkeley, CMU, Cornell, MIT, and Stanford, under the leadership of Bob Kahn and John Garrett of the Corporation for National Research Initiatives (CNRI), the prime contractor.

From the standpoint of a Dienst user, a document collection consists of a unified space of uniquely identified documents, each of which may be available in a variety of formats. Using publicly available World Wide Web clients, users may search the collection, browse and read individual documents in any of their available formats, and download or print a document. The Dienst system also provides site administrators with tools for managing their document collections, including, among others, automated submission procedures, indexing tools, and format conversion tools.

Dienst servers execute through gateways from any World-Wide Web (WWW) server that supports the Common Gateway Interface. Dienst protocol requests are embedded within HTTP, the WWW protocol. The Dienst protocol includes requests to search for documents, access one or more servers, display particular parts of documents in specific formats, print or download all or parts of documents, and the like. The protocol is designed to support both user-level and program-level queries, making it easy to write automated systems that access collections served by Dienst.

An implementation of Dienst is now being used by CSTR consortium members and a number of other C.S. departments to make their technical reports available over the Internet, providing access to thousands of computer science technical reports[1]. The technology that underlies Dienst is not specific to the computer science technical report domain and can be applied to many other document collections. We should note, however, that the current implementation does not address issues unique to copyrighted commercial materials, such as limiting access to authorized individuals and charging.

3 Experience Gained

3.1 Dienst and the World Wide Web

Dienst is implemented as a layer above the architecture of the World Wide Web. This allows Dienst to exploit the functionality of four components of the Web:

- HTTP protocol — Dienst protocol requests are encapsulated within the URL of an HTTP request. Dienst protocol responses are encoded as HTTP responses.

- WWW servers (HTTPD) — Dienst servers are accessed through gateways from any WWW server that supports the Common Gateway Interface (CGI).

[1]The Cornell Dienst server is accessible at http://cs-tr.cs.cornell.edu.

- HyperText Markup Language (HTML) — the Dienst user interface is implemented using HTML document markup, most notably the forms feature.

- WWW clients — the three preceding features make Dienst available from any full-featured WWW client.

This integration with the WWW sets Dienst apart from other systems that provide access to online documents. Traditional commercial systems, such as Dialog and Lexis/Nexis, while available over the Internet, are accessed via terminal interfaces. These systems are often difficult to use, because of limited user interfaces, and deliver only ASCII documents. A number of other systems have been developed based on client/server architecture. Notable among these are the System 33 Document Service [6] from Xerox, the RightPages system [7] from AT&T Bell Laboratories, and the CORE project [3] from BellCore, Cornell University, OCLC, and others. Each of these systems use proprietary clients, servers, and protocols. Finally, there are a number of systems that, while accessed through the World Wide Web, are mostly interfaces to central indexing sites; the actual documents are retrieved using URL's to FTP sites. Two examples of this are the Wide Area Technical Report Server (WATERS)[5] and the Unified Computer Science TR Index (UCSTRI)[9].

Dienst benefits in a number of ways from its use of the World Wide Web:

1. Browsers for the World Wide Web are available, generally for free, for all popular workstations and personal computers. The increasing number of users accustomed to the Web will have little problem using Dienst.

   Current Web browsers provide a number of powerful features. Three of these features are central to the Dienst interface. Browser support for embedded images is used to display document pages in Dienst. The ability of browsers to launch viewers, or helper applications, as a function of document type facilitates Dienst's multi-format document model. Finally, the imagemap feature, whereby the location of a user click in an image is transmitted to the server, is used by Dienst as an aid to document browsing, image zooming, and full-text searching.

2. WWW servers are available for many UNIX platforms and generally are free. The current mainstream servers, NCSA httpd and CERN httpd, provide a number of features that Dienst automatically inherits. These include clickable image support, ability to run as a proxy which provides support for firewalls, and the ability to control access to specific users or groups of users. We can expect that servers will soon include a number of additional features, such as tools for user authentication, payment mechanisms, and support for new graphics standards. The first two features are of importance so Dienst can be used where access to documents must be strictly controlled or cost recovery is important.

   Finally, WWW servers are now installed at a growing number of institutions with Internet access. System administrators are familiar with configuring and maintaining these servers. Running a Dienst server as a gateway from an existing Web server requires few additional administrative tasks.

3. HTTP provides a mechanism for transmitting meta-information such as file format (MIME type) and date separately from actual file contents. Dienst uses this to provide explicit types for documents that it returns, eliminating the need for client programs to guess the type.

4. The forms features of HTML permit the development of a common cross-architecture user interface. Although the user interfaces provided by forms are far from perfect, they are reasonably powerful, as demonstrated by the number of interactive services on the Web.

Dienst's dependence on the Web has some negative aspects. Browsing through multiple page documents does incur high overhead due to the stateless nature of HTTP. The Web has insufficient mechanisms for displaying documents with multiple textual and non-textual parts. We believe that compound documents, especially mixes of text and video, will become more common in the future. Servers are also not aware of the display capabilities of clients, making it impossible to "guess" at the best default format of a document. We expect that many of these issues will be resolved as development of Web technologies continues.

3.2 Copyright Issues

Traditionally, computer science technical reports have been distributed in hardcopy form with little concern for copyright. Frequently, there would be no explicit copyright notice on the publication, which before March 1989 effectively put the technical report in the public domain [8, chapter 4]. Moreover, authors would typically assign exclusive rights for a journal version of the same paper to another publisher, with both author and publisher ignoring the existence of a technical report version of the paper.

With the advent of Internet distribution of technical reports, concerns over copyright and intellectual property issues have been greatly heightened. When technical reports existed only as a relatively small number of copies with a limited distribution, journal publishers could afford to take the position that they would have little impact on their own sales and subscriptions. When an unlimited number of copies are immediately available over the network to any interested researcher, the impact on the traditional journal is much more significant.

In developing Dienst, whose purpose was to make access to Cornell's technical reports simple and immediate, we therefore had to deal with several issues related to copyright:

1. Technical report authors wished to ensure the rapid dissemination of breaking research results. Typical journal publication was a process that could require two or three years.

2. Since promotion and tenure decisions are based in part on publication in refereed journals, authors were not prepared to have the publication of a paper as a technical report endanger its future publication as a journal article.

3. Users of the technical report server deserved a clear statement of their rights in the reports they were accessing. Could they print a copy? Multiple copies? Could they e-mail it to another researcher? Could they permanently archive a copy for redistribution?

4. The legal status of technical reports distributed over the Internet was (and is) unclear. Does network distribution count as publication, or is it a "performance" of the work [4]? What happens when someone prints a copy? Does fair use apply to reports "performed" over the network?

After extensive discussions with authors, librarians, and lawyers, we devised a scheme for dealing with copyright on Cornell technical reports. The author explicitly licenses full rights to Cornell, including the right to sub-license these rights to others. However, the author retains the right to withdraw this license on sixty days written notice. The exact wording of the copyright notice on the technical report server is:

> You are granted permission for the non-commercial reproduction, distribution, display, and performance of this technical report in any format, BUT this permission is only for a period of 45 (forty-five) days from the most recent time that you verified that this technical report is still available from the Computer Science Department of Cornell University under terms that include this permission. All other rights are reserved by the author(s).

This approach has the following features:

1. If the paper is accepted by a journal, then the author can withdraw the report and turn around and offer full and exclusive copyright to a journal (although the paper will have been previously published as a technical report).

2. The author grants full rights to Cornell, including sub-licensing. This means that Cornell can grant explicit rights to anyone wishing to view, copy, print, or redistribute the report. However, these rights *time-out* after forty-five days.

3. Once the paper has been withdrawn and the rights have timed-out, no further distribution, “performance”, or viewing of the on-line version will take place. However, existing paper copies are not affected, and continue to be available on loan from libraries and individuals. Copies can also be made under the *fair-use* provisions of the copyright law.

The issues of copyright for electronically distributed works are still far from clear. The scheme we have proposed here is untested in the courts, and so is subject to the vagaries of future legal interpretations. It also leaves the publisher with a work that has potentially been much more widely distributed than would have been the case with paper preprints. However, since one of the main functions of a university is the dissemination of knowledge, the benefit seems worth the risks. So far, this approach has proved to be acceptable to all the parties involved. It provides a clear legal basis for Cornell’s electronic distribution of the technical reports, and it seems to give authors, publishers, and users a compromise that they can live with.

3.3 Document Submission and Management

Digital library technology will only propagate beyond the technologically savvy if such systems require minimal expert administration and do not burden authors with technical details. Many of the organizations that wish to publish documents digitally (e.g. government agencies, academic departments) have little in-house technical expertise. Authors are concerned with writing documents and getting them published. Submission to a digital library should require little more skill than using a word processor.

We have implemented in Dienst a set of tools that mostly automate the process of adding documents to and managing a digital library. These tools are integrated with the Dienst digital library server. Our goal when creating these tools was to make the digital library maintainable by a document librarian (DL) with low-level computer training. This DL serves four major roles: 1) as the general manager of the collection; 2) as the reviewer of document submissions, to protect against counterfeit document submissions; 3) as the clearing house for copyright issues; and 4) as the archiver of document hardcopy.

The author submits a document to the library in PostScript. Rather than working with a plethora of document formats from a variety of word processors, we determined that PostScript represents a *lingua franca* that can be generated from virtually all word or text processing systems. We recognize that there will be documents that can not be represented in this fashion, but estimate that the number will be very few and that techniques for managing them can be developed as the process matures. The author submits a document to the library by completing an HTML form that contains text fields for bibliographic data about the document. These fields are document title, author(s), pathname of the PostScript file, abstract, and submitter's e-mail address. The submitter can quickly complete this form by "cutting and pasting" text from the document source.

The document librarian, in the role of gatekeeper of the system, learns of each submission through an automatically generated e-mail message. No document actually enters the document database until the DL manually checks the submission. In addition, the DL acts as the legal gateway, ensuring that the authors complete a copyright release form that gives the department permission to make the document available over the Internet. When manual checking and copyright clearing are complete, the DL uses a simple command to assign a unique document identifier (the *docid*) to the document and signal that the document is ready for entry into the database.

The remainder of the submission process is fully automated. Software that is integrated with the Dienst server generates the bibliographic entry from the submitter's entry, checks the validity of the PostScript file, builds the actual database entry, and generates the page images for online viewing and browsing of the document, and the plain text version for full text indexing. Text extraction is performed by rendering the PostScript into images, then performing optical character recognition. Although this is more expensive than simply extracting the strings from the PostScript, it also produces a higher quality output, since the OCR system we use (ScanWorX from Xerox) produces paragraph bounding boxes, which we then use to perform paragraph- level full text search. Figures and graphs are of course lost, but are of no use for full text search in any case. Once in the database the document is searchable and viewable from any interoperating Dienst server.

It is sometimes necessary to withdraw a document from the library due to invalidation of the results in the document, replacement by another document, or submission of the document to a journal. Whereas the decision to withdraw is up to the author, withdrawal from the database is effected via a simple command from the document librarian. When a document is withdrawn, users are unable to see either bibliographic information or contents of the document. The document identifier remains associated with the withdrawn document.

While electronic document delivery is the *raison d'etre* of our system, we recognize that publication quality hardcopy is sometimes needed. The document librarian must be able to produce paper copy for archival storage and for people who do not have electronic access. For this purpose, we have developed an interface between the Dienst database and the Xerox Docutech publishing system [1]. This system can be used to produce very high-quality bound and covered hard-copy of any document in the library database.

Finally, the Dienst software includes an extensive document database checker utility. The document librarian can use this utility to scan the entire document database and determine the integrity of the document files. In the current release of the software this utility only reports integrity errors, and repairs have to be made manually. In future releases, we hope to automate the repair process.

3.4 Providing Documents in Multiple Formats

Long before we invented Dienst, many sites were distributing technical reports via anonymous FTP. Why not continue to use FTP? Unfortunately, an FTP server appears to the user much like a file system. The user just sees a collection of files, perhaps grouped into trees. If several files jointly constitute a document (for instance, the scanned pages), the relation is not explicitly marked in any way, rather the user must guess from such evidence as similarities in the file name. Not even the file type or format is marked. Related files may be grouped together into a common directory; this would be helpful, except for the lack of standard conventions for the grouping. Even though humans are often able to correctly guess the organization, this is not possible for programs that might want to access the collection. A file system is just too low level an abstraction for a digital library.

A system that provides a higher level interface must somehow understand the storage conventions used at the site. It can try to do this by applying heuristics (which UCSTRI does, to an extent), but file systems are so diverse and chaotic that this can't work for every case. A more reliable approach is for each site to provide definitions that map the common abstraction into the particular conventions of the site. This is the approach used by Dienst.

Each Dienst installation includes specifications that provide a mapping from a *docid* and format name into a local pathname at that site. Thus, users can address

Dienst documents in terms of their names and formats, and Dienst translates these automatically into locally meaningful pathnames. This mapping is divided into two parts. The first maps a *docid* into a particular directory in the file system. Dienst requires that all files associated with a document be stored in or under a common subdirectory. This mapping is expressed by a subroutine, which differs at each site. The second mapping takes a format into a file name, and is expressed declaratively.

The actual names of formats are not intended to be seen by end users. Instead, the declaration also includes a "pretty" name to be shown to the user, as well as some further text for explanation. Thus, for example, the "ocr" format includes a disclaimer that "text produced by OCR may have errors". Some commonly used formats are: *thumbnail* — a reduced size image intended for rapid visual browsing; *inline* — an image intended for inline display; *scanned* — a page image as produced by the scanner; *postscript* — PostScript; *ocr*— text as produced by OCR.

The format names are chosen to be useful in describing the intended purpose of the formats, not the actual encoding used, because there might be several alternative encodings capable of meeting a purpose[2]. For instance, an inline image is usually stored as in GIF format, but could as well be an X bitmap, or (on some browsers anyway) a JPEG image. There is no reason in principle why TIFF or JBIG encoding could not be used.

4 Ongoing Issues

4.1 Interoperability Among Heterogeneous Search Engines

Queries to the existing Dienst digital library may be expressed either as Boolean combinations of keywords in fields (author, title, etc.) or as full text searches. Both

[2]In an earlier version of Dienst, we encoded formats with MIME types. This was a mistake.

forms of search return a list of *docids*; with full text search the list is sorted by relevance. We anticipate that when Dienst is applied in other domains, librarians will add additional fields to the bibliographic records. For example, if one were to use Dienst for a digital art library, one might add a field specifying the nationality of the artist, or the medium used in the original work. At present, we use the bibliographic format defined in RFC 1357, but the Dienst query protocol can be easily adapted to support additional fields or new bibliographic formats.

We further anticipate that some collections might wish to use entirely new query languages, such as the Common Command Language (CCL) used by CMU's Mercury Server. The Dienst protocol is extensible to support other query languages (so long as they can somehow be encoded into a URL), but further work is needed to define the semantics of a search expressed in one formalism when the base records are stored in another. In some cases, the query languages can be fully translated, but in other cases the mapping will not be complete. In the incomplete cases, is it better to return an error code and no result, or is a partially correct search better than nothing?

More work is also needed in combining results from multiple index servers. Boolean queries produce unordered sets of documents, which can easily be merged, requiring, at most, removal of duplicates, something easily done given that each document has a unique identifier. But full text queries return sorted results, and these are more difficult to merge, as each list is sorted according to statistics of term frequency that are only meaningful in the context of a single collection, not globally across the entire collection. A related problem is merging results from "foreign" servers such as WATERS or UCSTRI, which come in the form of raw HTML. Unless we attempt to parse the HTML and extract document information, the best we can do is to simply list the HTML in a separate section of the display.

4.2 Heterogeneous Servers and the Dynamic User Interface

An unresolved issue is the extent to which the differing capabilities of each search system should be reflected back to the end user. For example, CCL offers a "proximity" operator which finds words within a specified distance of one another; and relevance feedback requires users to select from all returned documents those that are relevant. This causes a conflict for the future of Dienst's user interface. On the one hand, we'd like to make it simple and uniform; on the other, we'd like to provide access to more powerful features when available.

To make such features available requires a dynamic user interface that changes its appearance depending on the features provided by the search system. This would require that the Dienst user interface server query an index server to determine what features are available before returning a search form to the user. However, this conflicts with another goal of Dienst: to try and hide the existence of multiple index servers from the user. How can we indicate that some searching features may be only partially applied to the document collection?

Whatever decision we take will be difficult to implement gracefully in the WWW, because user selections are available to the server only after the user has completely filled out the form. In more tightly integrated user interface tool kits, every user action is returned to the server as it occurs, thus affording an opportunity to reconfigure the interface. The obvious solution would involve multiple rounds of user submission and the creation of a more targeted query form, but this would needlessly exasperate many users.

4.3 Reliability

As the number of servers increases, the chances decrease that all will be in operation when a search is requested. How can we make the entire collection useful in such an environment? There are two alternatives open, and we need to explore both. The first is to begin replicating index servers, to increase availability. We may be forced to do this anyway, if Dienst becomes more popular. Many useful Web services are now so heavily loaded that their response time has become unbearably long.

A second alternative is to add persistent search, which continues to attempt to contact non-responsive sites. A persistent search would allow partial search results on the available servers to be returned immediately and the rest of the search to be "held", pending the availability of the remaining index servers. The full results could then either be locally cached for later user access, or mailed to the user. This does have the potential to make the user interface more complex, which is not desirable.

4.4 Document Structure

Documents have both physical and logical structure. Physical structure is the division into physical pieces, e.g. pages, and is already handled well by Dienst. When a document is stored in paged form (e.g. when it has been scanned from paper) one can request a specific page in a specific format. In the future, as we begin to handle more media types, it may be useful to add additional physical coordinates, e.g. SMPTE time codes for video or rectangular subsets of pages; but these extensions should not be difficult to design.

Logical structure, on the other hand, is only barely present in the current Dienst protocol. One may request the "Body" of a document, but no finer level of logical addressing is supported. Further granularity can be added, but only by extending the protocol as a whole. This won't scale well, as it is likely that there will be many alternative schemes for document structure, and requiring modifications to the entire protocol is too cumbersome. What is needed is some kind of generic addressing, so that any document may have any desired (hierarchical) structure expressed in locally meaningful units. This will require a new message which returns the abstract structural schema for a document (which might be an SGML

DTD), and a query which can accept coordinates in any coordinate system. Thus for example, suppose one has a legal document stored in Dienst. One would first determine that the document is structured as numbered sections with numbered paragraphs within, and then ask for paragraph N within section M.

4.5 Technology Transfer

Dienst servers are currently running at all the CSTR consortium sites and at a number of other computer science departments in the United States. The technology is relatively mature and development of the server and protocol is continuing at Cornell.

We have an ongoing effort to disseminate the technology beyond its current reach. Our strongest effort has been in the computer science domain; we have contacted almost all other Ph.D. degree granting computer science department in the United States. However, Dienst is not specific to computer science technical reports. The technology can be used by any network of sites that wish to provide Internet access to documents that are freely available and have minimal copyright restrictions. This is an increasingly common goal of other academic departments, government agencies, and corporations. We have made some initial contacts in these other domains, especially government agencies, and expect to see the establishment of other Dienst networks.

The transfer of the technology beyond its research base has met a number of impediments:

- Many sites are justifiably wary of any product of research. They are suspicious of the "industrial strength" of the software. Other sites are not confident that there will be any support for the software once the research effort closes shop.
- Although we have made some effort to simplify the tasks of installing and

maintaining Dienst servers, more work is needed. Computer science departments usually have some savvy grad student who is interested in tinkering with new software. This is not true of other departments or agencies. Administering a Unix system, in itself, is an insurmountable task for many sites.

- Many institutions are increasingly wary of committing to any "latest and greatest" technology. They have already had the experience of dedicating resources to some technology, only to see it replaced months later by something else (e.g., the rush to Gopher just a couple of years ago).
- Cost recovery is often an issue even when documents have a history of being available for "free". Many institutions use a small handling fee to cover shipping of paper documents and skim a little off the top of this fee to recover the costs of administering a document library. Providing these same documents electronically eliminates shipping and paper fees, but it does not eliminate administrative costs and even incurs some new ones. Yet with no cost recovery model in digital library servers, the small income stream is lost.

These are some of the many technology transfer issues which need to be examined as digital library research proceeds over the next few years.

5 Future Plans

In addition to dealing with the problems described in the section above, we plan to continue developing Dienst to extend the functionality, scalability, and robustness of the architecture. In particular, we intend to address the following issues:

Multiple Domains Currently, while Dienst supports uniform access to a distributed collection of documents, each such collection is independent. The

Dienst servers accessing one information domain, say computer science technical reports, do not interact with the Dienst servers providing agricultural information from cooperative extension sites. A user is expected to deal with a server covering documents in a particular subject area. Search results from multiple collections must be obtained and combined by hand.

We plan to extend Dienst so that such server collections provide information about the subject matter and content of the collection. This information will be available to users through a *server-of-servers* to assist users in finding appropriate information resources.

Moreover, as the number of possibly relevant corpora and Dienst servers gets very large, full searches over all available information become prohibitively expensive. It then becomes critical to identify the "best" servers for any given search. We intend to investigate mechanisms for carrying out this selection and incorporate the most appropriate into Dienst.

Compound Documents Dienst currently supports a variety of document formats, at both the page and the complete document level. However, each instance of a format is assumed to be uniform over the document. We intend to extend Dienst to support true compound documents, where a single document may include such elements as text, sound clips, MPEG movies, high-resolution color graphics, program code, and scientific data sets.

User Studies Dienst servers are already being widely used to access a number of documents. Moreover, Dienst provides a number of different capabilities for browsing, searching, and viewing documents. By instrumenting Dienst servers, we can collect information on a range of *usability* issues related to document access. Some potential topics are: What is the trade-off between viewing documents on-line and printing a copy? What search facilities are most useful — full-text, boolean bibliographic search, or searching on specific fields? How do different levels of network connectivity affect the usability of the library?

The Dienst architecture provides both an excellent production system for accessing a distributed collection of documents, and a robust platform for future work in document publishing and dissemination. Over the next several years, we plan to continue extensive development and support of Dienst as an easy-to-use "drop-in" system for the publication of on-line documents. Such a system has a wide range of potential applications, ranging from government documents, to research reports, to product marketing information, to on-line documentation. By providing a uniform, integrated, and freely available system for document dissemination, Dienst can have a major impact on network publishing.

6 Acknowledgements

This work was supported in part by the Advanced Research Projects Agency under Grant No. MDA972-92-J-1029 with the Corporation for National Research Initiatives (CNRI). Its content does not necessarily reflect the position or the policy of the Government or CNRI, and no official endorsement should be inferred. This work was done at the Design Research Institute, a collaboration of Xerox Corporation and Cornell University, and at the Computer Science Department at Cornell University.

References

[1] Cornell Information Technologies, Ithaca, NY. *How to Use EZ-PUBLISH and the Docutech Printer at Cornell Information Technologies*, November 1993.

[2] James R. Davis and Carl Lagoze. A protocol and server for a distributed digital technical report library. Technical Report TR94-1418, Computer Science Department, Cornell University, 1994.

[3] Richard Entlich, Lorrin Garson, Michael Lesk, Lorraine Normore, Jan Olsen, and Stuart Weibel. Making a Digital Library: The Contents of the CORE Project. *Communications of the ACM*, March (forthcoming) 1995.

[4] Patrice A. Lyons. Interactive access to computer programs over telecommunications pathways: Some copyright implications. Submitted for publication as part of ASIS Monograph, The MIT Press.

[5] Kurt J. Maly, Edward A. Fox, James C. French, and Alan L. Selman. Wide area technical report server. Technical Report TR-92-44, Old Dominion University, 1992.

[6] Steve Putz. Design and Implementation of the System 33 Document Service. Technical Report ISTL-NLTT-93-07-01, Xerox Palo Alto Research Center, 1993.

[7] Guy A. Story, Lawrence O'Gorman, David Fox, Louise Levy Schaper, and H.V. Jagadish. The RightPages Image-Based Electronic Library for Alerting and Browsing. *IEEE Computer*, pages 17–25, September 1992.

[8] William S. Strong. *The Copyright Book*. The MIT Press, Cambridge, Massachusetts, fourth edition, 1993.

[9] Marc VanHeynigen. WWW page. http://cs.indiana.edu/cstr/search.