# A server for a distributed digital technical report library

James R. Davis
Xerox Corporation
Design Research Institute
502 Theory Center
Cornell University, Ithaca, NY 14853
davis@dri.cornell.edu

15 January 1994

**Abstract**

This paper describes **Dienst** – a Distributed Interactive Extensible Network Server for Techreports. Dienst is based on the World Wide Web protocol HTTP, and provides an object-oriented interface to a document model. The document model allows one to access the document as a whole or by named sub-parts and it supports multiple formats for the document. The server also supports a user-level browser with full-text search and rapid visual browsing based on "thumbnail" (very small page) images.

The Computer Science Technical Report (CSTR) project is an ARPA sponsored effort to create an online digital library of technical reports from the nation's top computer science universities. This paper reports on efforts to develop Dienst, a server and browser for this emerging library. Dienst should be applicable to other kinds of collections as well.

The server and browser are built on the framework of the World Wide Web system[1](and this paper assumes that the reader is already familiar with the terminology employed by that system). The main contributions of Dienst are

- providing an extensible document model with an object-oriented interface which supports the notions of multiple formats for the same document and document parts, and
- a browser supporting forms-based queries (over author, title, and abstract) and rapid visual browsing with "thumbnail" images.

Dienst is intended to provide Internet access to a distributed, decentralized multi-format, document collection. The collection is distributed and decentralized in that documents are stored on many different machines across the net, sometimes with more than one administration at each site. The documents are multi-format in that a given report might be stored in multiple forms of representation (TIFF, PostScript, GIF, plain ASCII etc).

Dienst supports two very different sets of users. The first class of users are researchers developing information access programs as part of the CSTR project. These researchers are themselves distributed (at Berkeley, CMU, Cornell, MIT, and Stanford) and use a number of different platforms. Dienst allows researchers to access the data more easily than through anonymous FTP, while also imposing a smaller load on the host machine. The second set of users are those who require an easily used browsing interface for searching the database. These users might be anywhere on the Internet.

Providing two interfaces to the same data takes extra work, but it is a necessity since people and programs differ so widely in the way they can accept information[1]. Programs require rigid, shared conventions; these conventions are unlikely to make an acceptable human interface. Attempts at compromise, such as the Unix shell, fail for both purposes: they are difficult for humans to understand and awkward to program. One can expect similar similar difficulties as people attempt to write automated programs to explore the expanding world of network information servers, most of which are oriented towards people, not programs.

# 1 Related work

A number of other systems provide access to online documents. First, there are the traditional commercial systems, such as Dialog, Lexis/Nexis and so on, some of which are accessible from the net, and a number of campuses now provide online access to their catalogs, (for a list see Hytelnet[2].) These systems are not suitable for the CSTR project for several reasons. First, the interfaces are in general not easy to use, and each system has its own idiosyncrasies. Second, the systems are closed - one can not access the underlying data for experiments in information retrieval. Third, they often deliver bibliographic records only, not the actual documents, and even when the document is available it is in plain ASCII only, with no provision for graphics.

Many Internet sites provide access to at least some of their technical reports via anonymous FTP, and therefore FTP access is the underlying technology for all the systems described below. The ubiquity of FTP argues for its use, but we find it to have a number of flaws, all of which arise from the fact that the abstraction it provides is that of a file system hierarchy, which leads to a number of disadvantages to those who wish to provide automatic search, indexing, or retrieval. First, each each FTP site has its own ad-hoc organization within the file tree. One site groups reports under author's last name, another

[1] A special thanks is due to Mitchell Charity of MIT for pointing this out.

[2] `http://www.cc.ukans.edu/hytelnet_html/START.TXT.html`

by year, still another by topic. One site stores reports as PostScript, with a README file giving bibliographic details, another has all bibliographic information in a master file, and a third has subdirectories of TIFF files. And so on. Second, there is no reliable way to determine the *contents* of files. File names, owners, and modification dates provide clues but no certainty. Sites often provide some form of an index, but again the lack of standards make these suitable for automatic indexing only with a heroic degree of effort (as in Marc VanHeyningen's Unified Computer Science TR Index[3] (UCSTRI), which attempts to parse site index files at approximately 100 sites. While the indexing program supports a number of different index file formats, and succeeds admirable with some, it can in no way hope to cover them all.). Third, there is no reliable way to determine the *format* of files. Most programs try to guess from the file suffix. These problems combine and become more acute when the document proper consists of a number of different files which must somehow be combined, as for instance when it consists of a number of scanned pages. There is no way to express the concept of "all the pages of this document, in TIFF, at 200 dots per inch".

Several systems provide access to document collections by indexing bibliographic records. For the most part, they use WAIS as an indexing and search engine, often accessed through the World Wide Web. (The UCTR index mentioned above apparently uses an independently developed search system not further described.) Monash University runs a server [4] which indexes a collection of 10,00 bibliographic records, of which 2,300 have abstracts. A search against this collection returns a bibliographic record but does not provide an online copy of the document.

The Wide Area Technical Report Server (WATERS)[5] is a distributed WAIS index of CS tech reports being developed by Old Dominion University, SUNY Buffalo, the University of Virginia, and Virginia Tech. Like the Monash system, WATERS uses a modification of the Unix `refer` bibliographic format. Searching this index return a list of matching documents in a brief form. Selecting one of these produces an expanded citation, with abstract when available, and with an active link to PostScript or TIFF forms when available. The WATERS project is unique in also providing a set of tools (Techrep[6]) for managing these bibliographic records.

The UCSTRI, Monash, and WATERS systems all provide a user interface for end-user searching of the document collection, but none of them are open systems in the sense of functioning as building blocks upon which other experiments may be conducted. Researchers can not access their document collections except as end users. The user interface in Dienst also offers a few features the other systems do not provide (multi-format support, visual browsing.)

[3] `http://cs.indiana.edu/cstr/search`

[4] `http://www.ncsa.uiuc.edu:8001/daneel.rdt.monash.edu.au:210/cs-techreport-abstracts?`

[5] `http://www.cs.odu.edu/WATERS/WATERS-GS.html`

[6] `http://uvacs.cs.virginia.edu/Techrep/doc/Techrep.html`

# 2 Dienst as open system

Dienst uses HTTP[7] as an alternative to FTP. A direct consequence of using HTTP is that Dienst provides explicit types for documents, eliminating the need for client programs to guess the type. HTTP provides a mechanism for transmitting meta-information such as file format and date separately from the actual file contents. Dienst builds upon HTTP to offer two additional features:

1. It presents a higher level abstraction, that of a document with multiple formats, instead of that of a file system, and this abstraction is uniform across all sites running Dienst.
2. It provides an object oriented interface to the document model.

Dienst models the document collection as a set of documents, each with a unique document identifier, or *docid* for short. The docid has two parts, a *publisher* which designates the institution which controls the document and a *string* provided by that institution. Each publisher has a unique name (although no mechanism for enforcing uniqueness currently exists), and each string is unique within a publisher. Thus the docid naming convention assigns a unique name to every document in the entire distributed collection. An example docid is `CORNELLCS:TR-92-1321`. This should allow one to refer to a document regardless of where it is stored (or indeed, regardless of whether it is online at all.) although the present implementation requires that one also know the correct server. (When the design of the Universal Resource Name conventions are complete, the docid naming convention will be brought into compliance with it.)

The document model is that each document consists of a set of named parts. The set of parts is extensible, and currently consists of the *Bibliography* and the *Body*. The bibliography consists of bibliographic information, and the body is the document proper. At some time in the future we may wish to further divide the body, e.g. into named sections, a table of contents, list of references, and so on. A second, orthogonal division of the body is into a set of numbered pages.

## 2.1 An object oriented interface encoded into URLs

Dienst supports an object oriented interface in that clients may make inquiries about the document collection or about individual documents by sending requests with properly encoded URLs. The messages currently supported allow clients to obtain a list of docids on the server, a list of all bibliographic information for all documents, or a list of all formats for a given document. The first two messages can return information about the entire collection or about subsets of the collection.

[7] `file://nic.merit.edu/internet/documents/internet-drafts/draft-ietf-iiir-http-00.ps`

The message-passing interface is distinct from the methods provided by HTTP itself (GET, PUT, etc), being implemented in the URLs of the protocol. It is made possible by designing the space of URLs accepted by Dienst to allow a URL to express, unambiguously, the name of the message, the docid it applies to (if any) and arguments to the method (if any). The resulting encoding is perhaps a bit more complicated than one would like but does suffice. So, for example, the URL `/Server/TR/`*docid*`/Formats`, where *docid* is replaced by a specific docid, returns a list of the formats the document is available in, one per line. Document formats are expressed as Mime[8] content types. Dienst is unique in employing content type parameters, allowing it to distinguish between similar types at multiple resolutions.

Some of our intended users will want to access Dienst by running Unix scripts on a periodic basis. To make their task easier, we also provide a number of small client programs, thus hiding the details of the URL encoding and the underlying HTTP protocol.

## 2.2 Implementing the document model

Dienst hides the underlying file system structure, presenting the appearance of a flat space of documents. To do this, it must be told how to map from this flat space into the file system structure. To install Dienst, one must provide a set of declarations with this information. For example, on the Cornell server, all files for TR-92-1321 would be stored in `/fsys/cs-tr/z/tech_reports/1992/92-1321`; and a GIF file at 721 dots per inch for page 4 would be stored in `0004.gif` in the `GIF-72-4` sub directory. Code fragments showing this mapping information are in figure 1.

```
sub docid_to_directory {
    local ($docid) = @_;
    local ($publisher, $series, $year, $number) = &Parse_docid ($docid);
    $document_dir . "/" . "19$year" . "/" . $year . "-" . $number;}

&def_storage ("TR", "image/gif;dpi=72", "GIF-72-4/%04d.gif", "page");
```

Figure 1: Examples of declarations of storage layout.

Note that these declaration permit, but does not require, the existence of several different document *series* (in this case, the series is "TR") each with its own storage conventions. This is useful when document file systems are not centrally controlled, and have divergent storage conventions.

[8] file://nic.merit.edu/internet/documents/rfc/rfc1521.ps

# 3 The Dienst user interface

In order to support people who want to search the document collection, we provide an HTML based World Wide Web interface. We assume the user has access to a suitable World Wide Web client (e.g. Mosaic, Lynx, Cello), and provide a forms-based interface for searching (Figure 2).

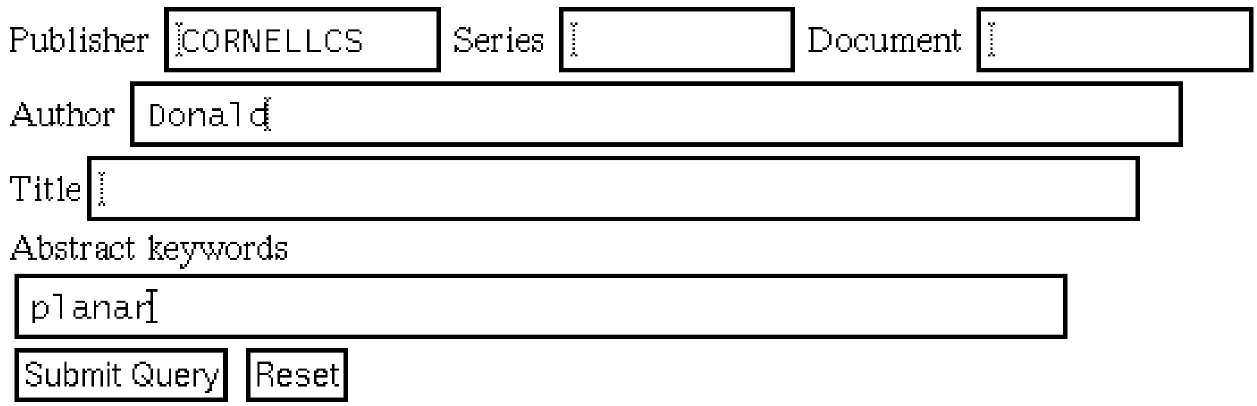


Figure 2: Search form for document collection

The user fills out the form, specifying an author's name and/or words from the title or abstract. Dienst returns a list of all records that meet the search criterion, in a brief form which includes the document id, title, and author. The user may select any of these, and see more detail about the document. The detailed report for a document shows the abstract, list of available forms, and a legal notice. This listing shows the abstract (when available) and a list of the available formats for the document. The document formats are of two classes, those for the entire body of the document as a whole (e.g. Postscript, plain text) and those which are page by page (e.g. TIFF).

Dienst supports online reading by displaying pages at 72 dots per inch, four bits per pixel. Most of the documents were captured by scanning a printed page at a resolution of 300 to 600 dots per inch. Such a scanned file is not suitable for online reading because it takes too long to transmit and display, moreover the extra resolution is otiose given that a typical display shows between 70 and 75 dpi. We therefore generated reduced resolution copies of all the page images in GIF format. An additional advantage of this format is that the Mosaic browser can display the page image inline, a convenience we appreciate.

Dienst supports rapid visual browsing by displaying reduced size ("thumbnail") images of all the pages, at 10 dpi (85 x 110 pixels). These images are far too small to read, but allow one to visually identify structures such as figures, equations, charts, and lists of references.

Figure 3: A thumbnail image

### 3.1 Bibliographic information

Dienst uses a record format developed for the CSTR project, specified in RFC-1357[9]. It is a simplified version of the MARC format, simple enough to be easily entered by clerical staff. (It would probably be possible to write translators between RFC-1357 and the other common bibliographic formats such as BibTex, or `refer`, and it will probably be necessary if Dienst is to have widespread acceptance.) At present Dienst uses an ad-hoc search engine which it allows it to search against the fields of the bibliographic record. It's clear that the search engine won't scale up to truly large collections, and we plan to switch to Smart[2] in the near future.

## 4 Discussion

Dienst is currently running at three sites: Cornell[10], UC Berkeley and Stanford, and is being studied by MIT and CMU. The Cornell server contains about 630 technical reports, and has page images for about half of these. It has been publicly available for 12 days, in that time is has been accessed at least once from 719 different hosts. It is too soon to tell whether Dienst will be successful, as it has only recently stabilized. At this point no researchers have based clients on it. It will also be very instructive to see whether other sites are willing and able to install it.

Already it is clear that some further work is required. First, given that a docid uniquely identifies a work, independently of the actual storage location, clients ought to be able to obtain a document from any Dienst server. If the document is not stored locally, the server can return a "Redirection" reply to the client with the name of the proper server. In order to do this, we need a way to locate a server given a publisher. We believe that the Domain Name Service is the appropriate means to do this.

The browser interface has been released only about a week, and has already seen widespread

[9] `file://nic.merit.edu/documents/rfc/rfc1357.txt`

[10] `http://cs-tr.cs.cornell.edu/`

casual inquiries from across the net, but only time will tell whether the interface is congenial enough and the contents valuable enough to widely used. In the near future, we plan to extend the search capabilities by offering full text search of the entire document using the Smart information retrieval system - a gateway between Dienst and Smart is under construction.

It is also clear that we must provide some kind of support for printing. Printing PostScript is already supported by existing WWW clients, but printing TIFF is not [11], and it is clear that we will have to add new method to Dienst to return the number of pages (so the client can retrieve them all) or a full tiff file for the entire document.

Finally, we must also consider issues of intellectual property rights. Negotiations on intellectual property were by far the largest delay in making the server available, and we consider technical reports to be among the least problematic documents in this regard, since we give them away free. It's clear that these problems will be much more challenging when we expand the document server to other publications.

## 5 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. I wish to thank my colleagues on the CSTR project for helpful discussions; especially Mitchell Charity, Dean Krafft, and Daniela Rus.

## References

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

[2] Gerard Salton. *Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.

[11] Regrettably, I don't know a convenient method for printing a TIFF encoded file. I would be grateful to hear from a reader who can provide one.