"Drop-in" publishing with the World Wide Web

Jim Davis and Carl Lagoze
Xerox Inc. and Cornell University

Abstract

> The goal of drop-in publishing is to simplify digital publishing over the Internet. We would like digital publishing of non-commercial matter (e.g. technical reports, course notes, brochures) be as easy as sending email is now, but with the virtues of archival storage and easy searching that we associate with electronic libraries. We propose a protocol, Dienst, to allow communication between clients and document servers by encoding object-oriented messages within URL's. A preliminary version of this protocol now runs at eight sites, and we describe some of its features. Next we present tools for automating the maintenance of document collections. Finally, we discuss the problems we've had with the Web as it stands, hoping to motivate changes that would improve performance of digital library systems such as ours.

A library with no limits...

> "However one may sing the praises of those who by their virtue either defend or increase the glory of their country, their actions only affect worldly prosperity, and within narrow limits....[but] Aldus is building up a library which has no other limits than the world itself."

Desiderius Erasmus wrote these words in praise of his friend Aldus, a book publisher of the 16th century. More than 400 years later, digital publishing may finally enable us to fulfill this vision, providing universal access to all the world's information. What's in the way?

The existing technologies (WWW, gopher, and even anonymous FTP) make reproduction and transmission fairly fast and cheap, but do little or nothing to help writers write or readers find or read documents. In our view, the problem is that they provide too little structure to the document collection. All of them present basically the same abstraction, namely a hierarchy of files, but do nothing to help the user locate a file within a hierarchy. Every site is different. Some group reports by year, others by project name; but even if every site on the Internet organized its hierarchy identically, it would not be enough, because every site also has its own conventions for naming files, indicating data formats, and making searchable indices. A writer who wishes to contribute has basically the same problem - it's easy to copy a file into an anonymous FTP area, but hard to make sure that it's indexed properly. A considerate writer might want to provide the same document in several formats, to increase the chances of accessibility, but this is a nuisance. We claim what's needed is a new, higher level protocol that hides the underlying details, and the underlying tools to simply library management.

This paper presents our first steps towards the universal library. We describe a protocol for universal access and the server that implements it. (For those familiar with our server - in this paper we describe not the currently running protocol, but rather the one we have submitted as an Internet Draft [DIENSTPROT] , which corrects a number of design flaws in the working version. We regret any confusion this causes.) We present a number of tools that integrate with our server to make publishing a document on-line relatively easy. We also discuss the steps we took to bring a large, existing collection online from paper. Finally, since our protocol is based on the World Wide Web, we also describe some of the problems we've observed in using it, in the hope that others at this conference will have solutions we can adopt.

Our focus on non-commercial publishing requires explanation. We realize that some content providers will not place their intellectual property on the net until clear definitions of legal rights and mechanisms for payment and protection are in place. We have nothing to contribute in these areas. Nevertheless there are a number of providers, such as universities or corporate internal groups, for whom these issues are less pressing, and we believe that we can thus make some useful contribution without working on the additional issues raised by economics.

Dienst provides a uniform protocol for document access

Dienst is a protocol for search, retrieval and display of documents. Dienst models the digital library as a flat set of documents, each of which has a unique name, can be in many formats (e.g., TIFF, GIF, Postscript) and consists of a set of named parts.

Dienst supports a message-passing interface to this document model. Messages may be addressed to every document server, to a particular server, to one document, or to a particular part of a document. A message is encoded into the "path" portion of a URL, and contains the name of the message, the recipient, and the arguments, if any. A message may be sent to any convenient Dienst server (the nearest, for example), which will execute it locally if or forward it as appropriate. Dienst appears to be a single virtual document collection, and hides the details of the server distribution. (Note that the actual implementation does not use an object oriented language, we use message passing only as a convenient conceptual model.)

Each document in Dienst has a unique identifier which names each document in a location-independent manner. This identifier, called a **DocID**, serves exactly the same role as a URN, and when URNs are fully specified we will adopt them. A DocId has three components: a **naming convention**, a **publisher** and a **number**. To ensure that each DocID is unique, each component (or rather, the institution that issues each component) guarantees that the next component is unique - thus each naming convention controls a namespace of publishers, and each publisher issues a set of numbers.

For each publisher, there must be at least one server to handle messages for the documents issued by that publisher. In our view, the minimum commitment a publisher must make to issue a document is to store and deliver the document to the network. When a Dienst server receives a message for a document it locates the closest server for the document's publisher and forwards the message to it.

Dienst messages address four types of digital library services: **user interface** services which present library information in a format designed for human readability, **repository** services, which store the document, and support retrieval of all or part, **index** services, which provide search, and **miscellaneous** services, which provide general information about a server.

Of these four services, only the first is used directly by a human. The others used by programs, in particular other Dienst servers, but also by other digital library or publishing systems. For example, the

Stanford Information Filtering Tool ([SIFT]) obtains bibliographic records through the index interface, and we are currently designing a gateway to the WATERS ([WATERS]) system. We encourage other developers of digital library systems to provide both user-interface and application-interfaces to their systems.

All services except the last are optional at a given site. This allows maximal flexibility in the way that particular server implementations interoperate. For example, one server may exist solely as a user interface gateway, providing transparent access for users to a particular domain of indexes and repositories. We see this flexible interoperability as key to the development of a digital library infrastructure where the "collection" will span multiple sites and continents.

Repository servers store documents in multiple formats

A key difference between Dienst and other current digital library systems is its ability to represent documents in multiple formats. Most current digital libraries present documents in exactly one form, PostScript. Although PostScript is almost always available for newly produced documents, there are problems with relying on it to the exclusion of all other formats. First, most older works are only available in paper, making scanned page images the only practical means of bringing the material online. (We describe our experiences in doing that below.) Second, looking forward we can expect to see other document representations become popular. (Surely at a World Wide Web conference we can claim that HTML will be used.) A third reason is that for some applications, other formats are just better. For example, if one wishes to do full text indexing on a document collection, the plain text is more useful than the PostScript file, and if one wishes to display just a single page, a collection of page images may be better than searching through PostScript. Therefore, Dienst's conceptual data model, allows each document to be stored in one or more formats.

The Dienst protocol includes a message that requests a document for a list of formats in which it is available. We specify formats with MIME ([MIME]) Content-types. Dienst does not support the notion of explicit conversion between document formats (as does System 33 [Putz]). A repository willing and able to provide a document in a given format should simply list that format, even if it is only obtained through a conversion service.

Diversity is the rule on the Internet, and each site supporting Dienst is likely to store their documents in a different way. The Dienst protocol hides all detail of the underlying storage organization -- this is in sharp contrast to FTP, Gopher, and "bare" HTTP, where the underlying hierarchy is visible. Each Dienst repository includes a function which maps from a DocID and format to the actual storage pathname on that server. This hides both details of file system structure and file typing or naming conventions from outside users. Thus one may request, say, the second page of the TIFF version of a document from a server without needing to know where and how it is stored.

Index servers support search

An index server accepts queries (in some query language) and searches for document records that satisfy the query. In our model, an index server is totally distinct from a repository. Repository data is likely to be huge, but index servers store only meta-data, which is quite modest in size. The choice of a query language is crucial to the power of an index server. As we did not wish to make this choice, the Dienst protocol is designed with one initial query language, and provision for extension to support others.

Every query language is based on an underlying model for the meta data it queries. The initial query language in Dienst assumes a minimal data model, where documents have an author, title, and abstract in addition to the publisher and number. A query may refer to any of these fields; if it refers to more than one then the terms are connected with an implicit "and". Thus one might query for all documents published by author "Wilson" at publisher "Stanford".

A search request returns a document of type `text/x-dienst-response`, consisting of records containing meta-information on all the matching documents. This meta-information follows the encoding proposed for Uniform Resource Characteristics (URC) [URC] . The URC draft proposes fields such as title, author and Content-type and URL, all of which which are obviously applicable; we have added a number of experimental attributes.

A prototype implementation runs at eight sites

An initial version of Dienst and a prototype implementation were developed as part of the Computer Science Technical Report (CSTR) project, an ARPA-sponsored, CNRI-directed effort to create an online digital library of technical reports from the nation's top computer science universities. This version was installed at the five universities that form the project (Cornell, CMU, Berkeley, MIT, and Stanford), and shortly thereafter at Princeton, Dartmouth, and Rochester. Here we describe a few of its features. A full account may be found in [Dienst].

One uses Dienst by connecting to any convenient Dienst server (that supports the user interface services) using a standard Web client. This server will display a form for searching the collection. Unless the user restricts the search to a single publisher, all Dienst servers are searched in parallel. Each Dienst server is made aware of all other Dienst servers by fetching a list of all servers from a single, central meta-server. Thus when a new server comes online, other servers become aware of it after only a short time. The results from a search are displayed as a list of the DocID, author, title, and date for each matching document, and include a URL for each document. Selecting one displays the document in more detail, including a list of the available formats (obtained as described above.) The user can retrieve the document in any of the formats.

Some repositories include page images as 4-bit 72 dot per inch GIF files. When this is the case, the user interface service is able to display the document page at a time, inline on the user's Web client. We found that such pages are readable on most monitors and saves considerable network bandwidth compared to the 600 dpi TIFF images. In addition, some sites also store reduced size "thumbnail" page images, which allow the user to quickly browse through a document and then click to view a interesting page (say one that contains a graphic) in full-page version. Although we do not have any formal user studies, anecdotal evidence says that this is a very powerful and helpful feature.

The server also allows the user to download and/or print all or selected pages of the document. Local users may print directly, while remote users can download a PostScript version of the document and then print it manually. Since all documents are not available in PostScript, the server has the ability to translate from TIFF images to level 2 PostScript on the fly.

Maintaining the Document Collection

Our goal is to simplify the process by which an author publishes digital documents. Much of the work in this area is at the document creation layer - that is, enhancements to HTML and/or HTML editors. Our approach is to allow authors to use their traditional text production system - LaTeX, troff, Word, etc - and then provide tools by which they can submit the results of that text processing to a digital library

Dienst simplifies digital library maintenance

Digital library technology will only propagate beyond the technologically savvy if such systems require minimal human intervention, especially by trained experts. Two points are obvious. First, authors are concerned primarily with writing documents and getting them published. Submission to a digital library should require little more skill than using a word processor. Second, many of the organizations that wish to publish documents (e.g., government agencies, academic departments, small companies) have little technical expertise. These organizations might tolerate the need for a reasonable skill level to install a digital library system (we intend to address the skill level required to install the digital library system in future work). However, they surely will not tolerate the cost of a systems expert to maintain the library.

At Cornell we have implemented a set of tools that mostly automate the process of managing a digital library. The tools are closely integrated with the Dienst digital library server. They are similar in spirit to those implemented for the Wide Area Technical Report Server ([WATERS]) system, known as Techrep, but whereas Techrep is designed to maintain the centralized index and unstructured FTP-based document repository that is characteristic of WATERS, the tools described here are tailored for the distributed indexes and structured repositories characteristic of Dienst.

Our design goal was to make the digital library maintainable by a document librarian (DL) with relatively 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. This system has recently been installed in the Cornell Computer Science Department and is now the means for all technical report submissions.

Authors add documents with an HTML form

The submitter prepares a document for submission by producing a PostScript representation. Rather than a plethora of document formats from a variety of word processors, we determined that PostScript represents a *lingua franca* that could 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 there number will be very few and that techniques for managing them can be developed as the process matures.

The author submits a document by completing an HTML form that contains text fields for bibliographic data about the document. These fields are the 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 validates submissions to the library

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 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 DocID to the document and signal that the document is ready entry into the database.

The remainder of the process is fully automated. Software that is integrated with the digital library server generates the RFC-1357 bibliographic entry from the submitter's entry, checks the validity of the postscript file, builds the actual database entry, and generates the GIF images for online viewing and browsing of the document.

The image conversions in this process are done with the Extended Portable Bitmap Toolkit ([PBMPLUS]). PBMPLUS consists of a number of filters for conversion between a variety of image formats (TIFF's, GIF's, X Bitmaps, etc.) and a small set of portable formats , and a set of tools to perform manipulations (rotations, color transformation, scaling) on the portable format files. PBMPLUS has the advantages of being free, quite reliable, usable on a wide variety of graphical formats, and quite powerful in its basic image manipulating capabilities.

Document librarian controls document withdrawal

A library system must be able to handle author requests for document withdrawal. The reason for withdrawal may be invalidation of the published research or newly published results in another document. For purposes of maintaining the integrity of collection, we have made the document librarian the control point for this operation. Document withdrawal, via a simple command, replaces the bibliographic file with an entry whose only attributes are the document number and a "WITHDRAWN" flag - all other bibliographic information is deleted. This ensures that the DocID is not reused for another document. Furthermore, the withdrawal moves the original bibliographic file and associated image and postscript files to a location that is not accessible to the document server.

Hardcopy is sometimes required

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 produce paper copy for archival storage and for people who do not have electronic access. In our system, printing of TR's is done using a package provided by Cornell Information Technologies called EZ-PUBLISH [EZPUB] . EZ-PUBLISH allows users across campus on various platforms to print to a central Xerox DocuTech publishing system. This is a publication quality printer that offers very high-speed and resolution (135 pages/minute, 600 dpi) and document setup facilities such as binding, different paper types, etc. With a command in the Dienst document management suite the DL can specify that multiple copies of a TR be printed on the DocuTech. The command does automatic setup of the print job including formatting of a standard Cornell Technical Report cover page.

We have just begun to use this automated system in the Computer Science department at Cornell. At a later time we will evaluate the effectiveness of the system, with special attention payed to the number of documents that require a special submission procedure (i.e., are not translatable to postscript). Obviously if the ratio of these is high to the number submitted documents, we need to rethink the design of the system.

Digitizing existing documents is a mostly manual task

We describe above a system for almost complete automation of the document submission process. At Cornell, we faced the additional task of converting an existing collection to digital form. While some of the tools described above were useful for this task, a large amount of manual intervention was required. The Cornell Computer Science Department has been publishing technical reports since 1968. As of September, 1994 the department had published 1449 TR's, with an average length of thirty-six pages (a total of over 52,000 pages). The digital record for many of these TR's is either non-existent, not easily available, or in a format that is difficult to interpret with current hardware and software (for example, a document formatted in an extinct copy of WordStar that is only available on floppies for a long-gone CPM system).

The common form that exists for all existing documents is hardcopy - the department maintains archival copies of the entire TR corpus. A production scanning facility on campus allowed the department to convert the entire corpus to high-quality 600dpi group 4-compressed TIFF images. Over a nine month period all hardcopy pages were scanned to individual TIFF files and downloaded via FTP to disk in the Computer Science Department. Each TIFF file ranges in size from around one kilobyte for a blank page to almost two megabytes for a page that contains a high quality photographic image. The total collection of pages images now occupies around 3.6 gigabytes.

It should be noted that scanning a collection, even as modest as the Cornell CS TR's, is time consuming, labor intensive, and not without problems. Even the most careful scanning technician occasionally misses pages, skews pages, or misses part of a page due to a unnoticed fold when the page is put on the scanner bed. These problems are difficult, if not impossible, to detect automatically. In addition, any problems that are detected are computationally intensive to correct. For example, a simple ninety-degree rotation of a 600 dpi TIFF image (due to incorrect scanning orientation) can take up to thirty minutes on a reasonably equipped SPARCstation 10.

An example illustrates the difficulty of correcting scanning problems. We discovered after all scanning was complete that many of our older TR's were scanned from pages that were oriented in landscape mode - two pages side-by-side. The result was a TIFF file containing two page images, which made correct page mapping impossible in the document server. While it was easy to find files with this problem (by reading the height and width from the TIFF header with a publically available TIFF package [Leffler]), reasonably quick correction required handcrafting c-code to split the files. Even with the handcrafted code, the location and correction process took over a week of compute time on a powerful workstation.

In addition to manual scanning of documents, we also had to manually enter the RFC-1357 bibliographic files. While it would have been easy to write translators between RFC-1357 and other common bibliographic formats such as BibTex, refer, etc, a consistent electronic bibliographic format was not available for all the TR's.

The Web is an imperfect document viewing technology

Basing our system on the World Wide has had both benefits and shortcomings. The obvious benefit is wide availability over publically available browsers. The shortcoming is that HTML, HTTP, and Web browsers lack a number of features important for digital document display and navigation. In this section we enumerate these features with the goal of inspiring discussion and enhancement of the technologies by the Web community.

Facilities for display of compound documents

The Web has insufficient mechanisms for displaying documents that consist of multiple textual and non-textual parts. In the electronic mail world, this issue is addressed by MIME (Multipurpose Internet Mail Extensions) [MIME]. Although HTTP uses MIME typing to allow browsers to map to the proper viewer for a document, documents are allowed to have only a single MIME type - multipart MIME is not one of them. The only facility for multi-format documents is the ability to embed images (either GIF or X Bitmap) in an HTML document. Yet there are gross inefficiency problems with image embedding since the HTTP browser must initiate an HTTP GET message for each embedded image. For a document with many embedded images, this can lead to unacceptable document download times. Furthermore, there are other types that one might like to embed in documents; for example, `MPEG` clips.

Ability to display in-line TIFF images

Among the many digital image formats (GIF, JPEG, PBM, etc.), TIFF is the most flexible and extensible. The TIFF specification is constantly evolving with the latest being Revision 6.0, finalized in 1992 [TIFF] . The most significant evolution, from the standpoint of reducing network bandwidth in image transfers, is the growing number of compression schemes for TIFF images. The ability to display in-line TIFF images in HTML documents would take full advantage of this rich de-facto image standard and permit immediate display of images produced by scanners, fax machines, and most paint and photo-retouching programs without computationally expensive conversion to GIF format.

Arbitrary rectangle section from the client

Viewing of document images on the Web would be greatly enhanced if the user of a client were able to select across an arbitrary rectangle in the image and transmit the selected coordinates back to the server. The server could then retransmit a "zoomed" image of the selected image, if the higher resolution were available (which it often is in a high resolution TIFF image). Image zooming is an important feature when the image being viewed is a document page that contains figures or tables with small fonts.

Client feedback on display capabilities

A main contribution of Dienst is that it supports the notion of multiple formats for the same document. The user can select among the available formats and use the view appropriate for that format. We would prefer to, at the server end, chose the "best" format to display on the respective client. This would be possible if the client HTTP request contained information on the display capabilities of the client system, especially display depth and size.

Authentication

The ability to restrict who is able to access a document is an essential feature of a production digital library. While our system is intended for non-commercial publishing, limiting access is required even in this domain; say, for example, documents that should only be read by members of a campus community or employees of a corporation. To do this, we require that server be able to guarantee the identity of those making protocol requests.

Summary

We have described a system, Dienst, that simplifies document publishing on the Internet. This system makes two important contributions.

First, Dienst provides a uniform protocol for search, retrieval, and display of documents. This protocol addresses a flexible document model where each document has a unique name, can be in multiple formats, and consists of a set of named parts. These parts can be physical, such as pages, or logical, such as chapters and tables. In addition, the protocol allows full interoperability between distributed digital library servers. The result is that the user sees a single virtual document collection.

Second, Dienst provides a set of tools that permit easy management of a digital library. These tools automate document submission, permit a document librarian to manage the collection, and facilitate the production of archival hardcopy.

We plan over the next year to build on this technology in a number of ways. Installation of the digital library server is too difficult. We intend to implement tools that will "auto-configure" the server. The search engine in the current implementation is primitive. We intend to include more advanced search engines, for example full-text search, to make document discovery in a collection more powerful and easier. The current strategy of conducting a parallel search over all servers does not scale over a very large number of servers. We intend to use meta-information about individual document servers to improve the search strategy. With this facility, one could, for example, choose to search only those libraries that have a high probability of containing computer science documents. We plan to examine and possibility incorporate current work on copyright servers, so Dienst might be used for commercial documents. Finally, we hope to use some of the current work in location-independent identifiers to refine the method by which documents on the net are addressed in Dienst.

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

[Cohen] Danny Cohen. A Format for E-mailing Bibliographic Records RFC-1357

[Dienst] James R. Davis, Carl Lagoze. A protocol and server for a distributed digital technical report library. Cornell University Computer Science Department Technical Report 94-1418, June 1994.

[DIENSTPROT] James R. Davis, Carl Lagoze. Dienst, A Protocol for a Distributed Digital Document Library. Internet Draft.

[EZPUB] Cornell Information Technologies. How to Use EZ-PUBLISH and the Docutech Printer at Cornell Information Technologies. November 24, 1993.

[GLOSS] Luis Gravano, Hector Garcia-Molina, Anthony Tomasic. The Efficiency of GLOSS for the Text Database Discovery Problem. Stanford University Technical Report CS-TN-93-2.

[Leffler] Sam Leffler. Public TIFF package. Available via ftp from sgi.com/graphics/tiff/v3.2beta.tar.Z .

[MIME] Nathaniel S. Borenstein, Ned Freed. MIME (Multipurpose Internet Mail Extensions). RFC-1521 .

[PBMPLUS] Jef Poskanzer. Extended Portable Bitmap Toolkit. Available from many anonymous FTP sites including ftp.ee.utah.edu.

[Putz] Steve Putz Design and Implementation of the System 33 Document Service Xerox PARC P93-00112, 1993

[SIFT] Online service at http://sift.stanford.edu.

[TIFF] Aldus Corporation. TIFF Revision 6.0 Specification.

[URC] Michael Mealling. Encoding and Use of Uniform Resource Characteristics. Internet Draft.

[URL] Tim Berners-Lee, Uniform Resource Locators (URL) Internet Draft.

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

[WATERS] Kurt J. Maly, Edward A. Fox, James C. French and Alan L. Selman. Wide Area Technical Report Server. Published online in http://www.cs.odu.edu/WATERS/WATERS-paper.ps

Biographies

Jim Davis works for Xerox at the Design Research Institute, a non-proprietary consortium at Cornell University which which seeks ways to improve the engineering design process. He received a PhD in 1989 from MIT's Media Technology Laboratory. His thesis, the *Back Seat Driver* was a computer program which provided spoken driving instructions to the operator of a car in real-time. Prior to that, he worked in research and development at a number of places including Atari's Cambridge Research Laboratory. At the DRI he works on developing electronic corporate memory. His most recent project is a system for shared group annotation using the World Wide Web. He also plays electric bass and is learning Dutch.

Carl Lagoze works for the Computer Science Department at Cornell University as a Senior Software Engineer in the CSTR project . He received a Master of Software Engineering from the Wang Institute of Graduate Studies in 1987. After receiving his degree he worked in both academia and the commercial world developing tools for the generation of language-specific editors. Over the past two years he has discovered the joys of digital libraries and the fascinating world of information capture and access. From the view of his non-technical friends, he is doing "something on that information superhighway." Mr. Lagoze is also the proud parent of the cutest baby ever and an avid cyclist and canoeist.

contact author: davis@dri.cornell.edu 607-255-1134