Dynamic Link Inclusion in Online PDF Journals

Steve Probets[1], David F. Brailsford[2], Les Carr and Wendy Hall[3]

[1] Electronic Publishing Solutions, Navdeep Court, Melton Road, Nottingham, UK, sgp@cs.nott.ac.uk

[2] Electronic Publishing Research Group, Department of Computer Science, University of Nottingham, Nottingham, NG7 2RD, dfb@cs.nott.ac.uk

[3] Multimedia Research Group, Department of Electronics and Computer Science, University of Southampton, Southampton SO2, lac@ecs.soton.ac.uk

Abstract. Two complementary *de facto* standards for the publication of electronic documents are HTML on the World Wide Web and Adobe's PDF (Portable Document Format) language for use with Acrobat viewers. Both these formats provide support for hypertext features to be embedded within documents. We present a method, which allows links and other hypertext material to be kept in an abstract form in separate link databases. The links can then be interpreted or compiled at any stage and applied, in the correct format to some specific representation such as HTML or PDF. This approach is of great value in keeping hyperlinks relevant, up-to-date and in a form which is independent of the finally delivered electronic document format. Four models are discussed for allowing publishers to insert links into documents at a late stage. The techniques discussed have been implemented using a combination of Acrobat plug-ins, Web servers and Web browsers.

Introduction

Over the past few years two very different developments have opened up the market for portable electronic document technologies. One of these is the World Wide Web (WWW) with its HTML markup language; the other is Adobe Acrobat where the underlying Portable Document Format (PDF) [10] is based on Level 2 PostScript. For the moment, these are the two major *de facto* electronic document standards but neither of them provides a comprehensive solution to the problems of producing, disseminating and indexing a distributed corpus of electronic documents. Instead their strengths and weaknesses are almost completely complementary which serves to highlight, yet again, the problems of bridging the gap between the "*document structure*" approach of HTML and the "*document appearance*" starting point of Acrobat's PDF.

This paper explains why dynamic link inclusion is beneficial and how it is achieved in two disparate standards such as HTML and PDF. HTML and PDF differ, not only in syntax but in their underlying link methodology. The paper goes on to explain how dynamic link inclusion can be achieved in PDF files by extending the functionality of the Acrobat viewer and how users can access the dynamic link inclusion software using the World-Wide Web.

World Wide Web documents are marked up in hypertext markup language (HTML) which is derived from SGML. There is no notion of a conventional 'page' nor of fixed

hyphenation and justification decisions. In a sense, therefore, HTML documents (like SGML ones) are driven 'top-down', from their structural tags, with detailed appearance on the screen being a function of the viewer rather than any inherent rasterising model in HTML.

Acrobat viewers, by contrast, are driven 'bottom up' from PDF, which is specifically designed for fast, on-screen, rendering of documents. The PDF document model is still strongly allied to pages and any resizing of the screen window in Acrobat leads to a cropping of the page on the screen, rather than wrap-around and re-formatting. There are simple hypertext features in PDF including links, bookmarks and electronic 'yellow stickers'. Acrobat also has an extensive application programmers interface (API) which enables developers to write code that can be executed under certain circumstances by Acrobat viewers. The programs developed by the API are known as *plug-ins*; they are not stand-alone programs but dynamically linked libraries, which are accessed from the Acrobat viewer. In this respect Acrobat viewers have similar extensibility to WWW viewers such as Netscape and Internet Explorer (which also have plug-in capabilities). Recent developments with Acrobat 3.0, such as a closer relationship with web browsers and support of forms, have underlined the importance of WWW for disseminating documents whether they be in HTML or PDF.

Despite the popularity of the HTML and PDF formats they are far from being the last word in electronic document evolution. We shall set out, in the next section, the current arrangements for embedding hypertext links into PDF and HTML documents, finding that the very notion of 'link embedding' constitutes a large proportion of the problem with respect to updating and maintaining HTML and PDF documents.

The goal of our research is to follow well-established principles from previous hypertext systems [5] [4] [13] which show that 'separable hyperstructure' confers enormous benefits in the flexible usage of an electronic document. By keeping hypertext items in a separate link database we are able to engineer a publisher's toolkit that enables hypertext features to be overlaid, as needed, onto documents held in a wide variety of formats. The very popularity of HTML and PDF make them the obvious candidates for testing out these ideas.

Link Implementation in HTML and PDF

In the HTML format, links are embedded in documents by an extension of the tagging structure used for features such as paragraphs, font changes and so on. For example a construction such as:

```
The <A HREF="http://www.w3.org/">W3</A> consortium ...
```

will cause the phrase 'The W3 consortium ...' to be rendered on the screen, with the word 'W3' being highlighted (usually in blue) by virtue of its position between the `<A>` and `</A>` tags. This highlighting denotes that W3 is to be the 'source anchor' for a link, with the 'destination anchor' being specified by the string beginning `"http://...`. Notice that this notation specifies very clearly *where* the destination document is held, via the Internet address `www.w3.org`. Once that server has been contacted the ultimate destination is the start of the default document at this address. This notation has the virtue that the word 'W3' remains the source anchor regardless of whereabouts on the

screen it appears but the hard-coding of the link destination causes all kinds of problems if the physical location of the destination document changes or if the destination file were to be renamed.

If embedded HTML links are far from satisfactory then things become even more desperate when we turn to PDF. The links can be intra-document (i.e. from one place in a document to another place in the same document), inter-document (i.e. from one place in a document to a particular place in another document on the same file system) or they can point at resources on the internet (web-links). The word 'place' in the previous sentence has to be taken all too literally: the source and destination anchors of a PDF link have to be specified as bounding boxes of the desired 'hot areas' on the appropriate PDF pages. A PDF version of the previous example could not tag the word `W3`, in the abstract, as being the source anchor of a link. Rather it assumes that, by one means or another, the position of the word on the page, and its bounding box, can be calculated so that a position-specified anchor can be created. To be fair, a small amount of indirection is permitted, for the destination of a link, in the latest version of PDF. But in quoting a label such as `FRED` for the destination of a link we still cannot escape from the fact that `FRED`'s absolute position coordinates and bounding box still have to be specified somewhere in the PDF file.

It is clear, then, that PDF links and bookmarks are hard to specify correctly. One way out of the dilemma is to generate these items during the typesetting process. This is achieved by inserting PDFMARK procedure calls into the PostScript file produced by the chosen text-formatting software. The PDFMARKs are PostScript procedure calls which specify hypertext items. When the PostScript is converted to PDF by Adobe's *Distiller* program, the PDFMARKs are mapped to the appropriate PDF hyperstructural item.

The CAJUN [15] project at the University of Nottingham used this technique to automate link insertion into documents. It must be noted that Hyperfeatures produced by this method are embedded into the PDF file as the file is created and, as such, suffer all the drawbacks mentioned in the next section.

A final point to note is that, like Acrobat, web servers are extensible, in that they provide hooks through which the functionality of the system can be extended. Whereas the standard way to add functionality to Acrobat is via plug-ins, web servers use Common Gateway Interface (CGI) scripts. A CGI script is a script or program that runs on the web server with end-users uploading data to the server (typically using HTML forms). The CGI scripts can access the data supplied by the form document and act on it accordingly. Web search engines use a form and CGI script interface. The CGI scripts can, if necessary, invoke Acrobat and utilise Acrobat plug-ins thereby enabling PDF files to be manipulated over the internet.

Separable Hyperstructure

The previous sections have shown that there are several interconnected issues to be addressed when developing a system for hypertext items that makes them easy to maintain and to update. These issues are the *separability*, *generality* and the *level of abstraction* that a document model permits its links (and other hypertext items) to possess. Turning

first to the embedding of links, we see at once that this approach makes it very difficult for documents to be shared in a useful way. Suppose we have a large, 50 Mbyte, PDF file with some publisher-supplied links already embedded within it. Suppose, furthermore, that this file is to be viewed by 50 different people in an organisation and all of these people want to add their own hyperstructure. The only way to do this, at the moment, is for each user to add extra links via Acrobat Exchange and to save a personal copy of the file with these new links embedded in it. The net effect of this is to create 2.5 Gbytes of extra PDF files, all of which have the same underlying imageable material and differ only in the hyperlinks that have been added. But if these extra links could be saved in a separate link base then a plug-in extension for the Acrobat viewer could interpret them and overlay them onto the imageable content supplied from a single master PDF file. This model bears strong similarities to the way in which re-entrant, or 'pure code', programs are administered in multi-user virtual memory systems. A master copy of the program code is shared between all users on the machine but each user has a separate and personal data segment. An incidental benefit of separated link bases is that these can be kept fully up-to-date quite independently of the underlying material to which they refer.

We have already described how a PDF link from a phrase such as `HELLO WORLD` is specified by the bounding box of that phrase on a given page. Although the Acrobat Exchange viewer does not allow the underlying material to be edited in any way, it does allow for pages to be added, deleted or replaced. The effect of replacing a page whose original version contained the source or destination of a link is to transfer the link to the replaced page with exactly the same position and bounding box. Thus, if the phrase `HELLO WORLD` has migrated due to a reformatting of the page then the anchor for the PDF link will *not* migrate with it. This effect underlines the case for abstractness in link specification and a late-binding implementation policy. If the specification of the source for the link, in the separate link base, specifies the anchor as being the first occurrence of `HELLO WORLD` in section 3.0 of a given document then the Acrobat plug-in can locate the page position of that anchor at the last possible moment thereby avoiding the problems of 'hard' embedded links becoming out-of-date.

Separable hyperlinks have been argued to be advantageous for many reasons, ([3][14] and [6], the latter based on the Dexter Hypetext Model [9]). They allow the author or reader of a document to create a personalised overlay of hypertext structure that is (to some extent) insulated from changes made to the source document's data or formatting. They also enable *one to many* type links to be specified (although how these links are implemented within a product like Acrobat must be addressed — see later). Finally, they provide an advantage in the case where the *linked-to* document is subject to change. A database of links makes the requirements for consistency explicit with respect to a set of both source and destination documents, allowing an authoring environment to check document editing activities against the explicit collection of link objects. For example, the Hyper-G environment provides strict guarantees of link consistency by monitoring all document changes and renames, automatically updating the relevant link objects as necessary, even if they are maintained on a different host computer to the edited document.

Let us look briefly at some existing hypertext systems, to see how they cope with the points raised.

Web Links

In order to overcome the problem that Web links are too specific in pointing to *where* information is kept rather than *what* the information is, the IETF Uniform Resource Identifier Working Group (`http://www.ics.uci.edu/pub/ietf/uri/`) was set up to investigate how link destinations could be specified in a more generic manner. Although the IETF Working Group has investigated ways to provide a 'resolution-mechanism-independent architecture for Uniform Resource Name (URN) usage and name space management' which would go some way to solving this problem, the likelihood of their results being implemented soon is doubtful. URN's should encompass scope, uniqueness and persistence using a resolvable extensible architecture. To achieve this goal numerous schemes have been developed by the IETF's working group. In the discussions that follow, links to web pages have been specified using universal resource locators, however there is no reason why URN link specifiers could not be used instead.

Acrobat Links

Link annotations within Acrobat differ from the link anchors found in web documents in that they link from and to specific views of a page, rather than from and to specific words within a document. Acrobat links are more closely related to the idea of a web imagemap in that link sources are specified in page coordinates whereas the web's HTML anchors link from a word or group of words to another document, or to a specific point within another document. Thus a textual description of an Acrobat link could be described as:

```
Link from the area enclosed by the coordinate pair (x1,y1)
(x2,y2) to file XYZ.HTML and display this document with
the anchor DESTINATION at the top of the browser window.
```

As mentioned earlier, in order to extend the lifespan of web links within PDF there is no reason why URN based schemes could not be adopted to work with the web links of Acrobat.

HyTime, Hyper-G and Microcosm

Other hypertext environments promote the idea of hyperlinks as first-class objects, managed separately from the documents to which they refer. The HyTime [12] standard defines independent links which (through a sophisticated combination of address resolution techniques) can point to the data at the source and destination of the link, while being stored in an entirely different location. Through the use of notation specific locators even data in highly specialised formats (such as PDF) can be represented in hypertext links, although, for reasons of simplicity, we chose to describe links using an SGML-based notoation (see Fig. 1).

Hyper-G (and its commercial form, HyperWave) [13] take advantage of independent databases of bi-directional links to provide guarantees of consistency across its global

hyperstructure, thereby eliminating the dangling link problem and alleviating some of the source/destination inconsistency problems that can arise when editing hypertext material.

In both Hyper-G and Microcosm [4] it is the responsibility of the document viewers to actively interrogate the link databases to discover any available links; other software which shares the same API may also manipulate links in various ways. For example, Microcosm provides a chain of software processes (called filters) which can handle the link requests on behalf of the viewer, expanding or pruning the selection of links that was originally requested. Microcosm's links themselves are of three main types depending on the context that a link is applicable to: a specific link is applicable only in one specific location in one specific document whereas, at the other end of the spectrum, a generic link is applicable at any point in the document corpus that a particular keyword occurs.

It has been argued that a separate database of links and documents is equivalent to a single database of documents with embedded links [1], however this is not the case when a link may be anchored at many places in the document corpus. In addition a separate link database is a necessity when dealing with a document format that does not itself support embedded links (e.g. plain ASCII text files or GIF images).

Four Models for Link Services

The advantages provided by separable databases of links have been outlined above. The form of this database is variable, although for this work we have a chosen a link database that can be described by a simple SGML DTD (an example of database entries can be seen in Fig. 1). The database consists of a sequence of links, and each link is composed of a source, a destination and some optional descriptors. Both the source and destination are described as a triple (document URL, offset within document, selected object within document) and allow the system to pinpoint the link anchors either by measuring from the beginning of a document (using the offset), or by matching a selection, or both.

The idea of external linkbases (Fig. 1) is very attractive in web-oriented environments where the rapid expansion of available data, means that hard-coded links in web pages and PDF articles (even using URNs) are very hard to keep up-to-date. External linkbases are much easier to maintain and manage; information has to be updated in one place rather than in multiple files.

The linkbases used in this work are derived from those used by Microcosm. In Fig. 1, the 'type=generic' attribute specifies that the link being described applies to all occurences of the source term in all files. Like Microcosm links could be restricted to specific files by specifying the source document ('type=local') and to specific words within specific files if the offset is specified ('type=specific'). Due to the nature of the PDF language, offsets cannot easily be calculated, however word occurence could be used instead (e.g. link from the 2nd occurence of the phrase 'Computerized Braille Typesetting'). By 'typing' links in this way source anchors can be precisely specified.

If external linkbases are used, the ability to dynamically include links in documents at the time the document is requested means that end-users can always be supplied with current links to the latest information. If linkbases are to be used as the source of hyper-

```
<link type=generic>
<src>
<doc><offset><sel>Typesetting
<dest>
<doc>http://cajun.cs.nott.ac.uk/wiley/journals/
epo/pdf/volume1/issue2/ephxj012.pdf
<offset><sel>
<title>Computerized Braille Typesetting: Another
View of Mark-up Standards
<link type=generic>
<src>
<doc><offset><sel>SGML
<dest>
<doc>http://cajun.cs.nott.ac.uk/wiley/journals/
epo/pdf/volume2/issue1/epdxb021.pdf
<offset><sel>
<title>Why Use SGML?
```

Fig. 1. An example linkbase

features in journal articles then there are various models that can be used to accomplish this. Different models occur when the linkbases and papers are in different locations and the model of choice may depend on who owns the papers and the linkbases. These models are described below and outlined in Fig. 2.

1. Microcosm model. The conventional Microcosm model assumes that publishers have control of articles and that users maintain their own linkbase files. In this instance, the user obtains the paper from the publisher and applies his links to the file to create a customised hyper-linked file.
2. Publisher control model. A second model occurs when the publishers maintain both articles and linkbases. Users request an article from the publisher and inform the publishers that they require one particular linkbase to be applied to the file. Publishers then include the links into the file and disseminate the file to the user.
3. Split control model 1. This is a model where publishers can make revenue from hitherto unforeseen areas. The publisher can allow (and charge for) their linkbases to be incorporated into files that end-users own (they may even have been purchased from other publishers). In this case the end-user sends his article over the Internet to the publisher, with a request that a certain type of link be included in his file. The publisher performs the inclusion and sends the file back to the user. This facility could be charged for, but, even if it is provided as a free service, publishers would not be losing out, as they could include links in their client's file that point to files that the publisher is selling. As an example of this, a researcher may own a PDF file about the use of radioactive isotopes in elucidating fish diseases. The main area of interest for the researcher may be in radioactive isotopes or it may be in fish biology. Assuming that the interest lies in fish biology and that the researcher knows that XYZ Publisher has an extensive fish biology linkbase, then the

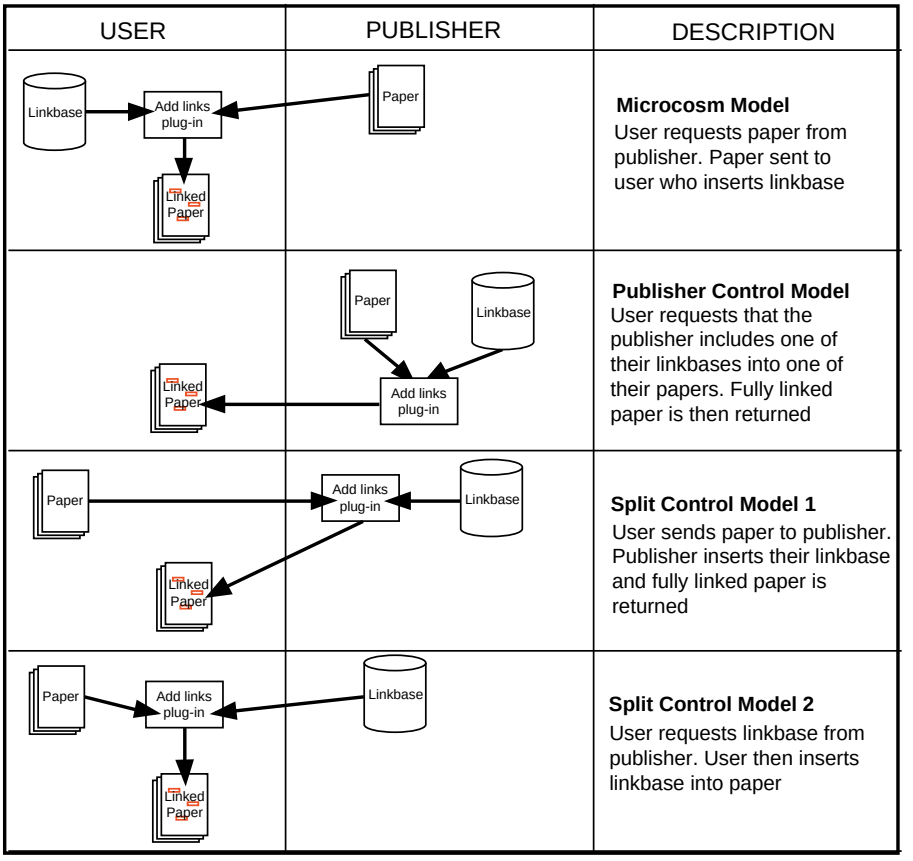

Fig. 2. Four models of link inclusion

researcher can transfer the paper over the internet to XYZ Publisher, with a request that the contents of the fish biology linkbase be inserted into the paper. Alternatively if both link subjects are of interest, then links from the fish biology linkbase could be inserted in one colour and links from the radioactive isotope linkbase in another colour.

4. Split control model 2. An alternative model which is similar to the previous model involves transferring the linkbase from the publisher to the client and enabling the client to incorporate the linkbase into any article maintained on the client's machine. The advantage of this model is that the majority of processing is performed on the client's machine (this model has been termed the heavyweight-client model) thereby reducing the load on network servers. The obvious disadvantage however is that publishers are distributing a saleable asset, the linkbase, into the community where it can be reproduced, transferred or altered without the publisher's consent.

The Open Journal's Acrobat Plug-in Toolkit

The Open Journal Project is a project undertaken by the Universities of Southampton and Nottingham in conjunction with a number of academic journal publishers [8]. The aim of the Open Journal Project is to provide a framework for publishing journals in a network environment such that maximum access to (and from) the publications is ensured. Part of the work performed for the Open Journal Framework (OJF) is to develop tools which enable the first of the split control scenarios to occur with Acrobat PDF files. Other parts of the project have enabled similar facilities to occur within HTML documents and documents in other formats. However, PDF appears to have become the standard format for electronically disseminating print-based journal articles, therefore the remainder of this paper deals solely with Acrobat PDF article dissemination.

In order to make PDF documents compatible with external linkbases, Acrobat plug-ins have been developed to incorporate links from a linkbase into a PDF file, and web compliant programs using the HTTP protocol have been developed to ease the transfer of PDF articles over the Internet from client to publisher.

A number of tools have been developed as part of the OJF project that will be directly of use to publishers of PDF products. Some of these tools are designed as tools to be used by publishers to include links into PDF documents before making the documents available over the Internet. Other tools are designed so that users can specify which types of links they require in files that they are downloading. The tools developed so far include:

- Basic OJF plug-in. This plug-in reads a linkbase and includes the links into a PDF file. If there are multiple links in the linkbase with the same source, only the first link is included. The included links follow the same specification as that required by Adobe's *weblink* plug-in, therefore PDF written with this plug-in can be read with the basic Acrobat Reader, or Exchange with the *weblink* plug-in.
- Multiway OJF plug-in. This plug-in also includes the links from a linkbase into the PDF. It differs from the basic plug-in in that if a source word has multiple links emanating from it (e.g. if there are multiple entries in the linkbase for particular phrase), then all these links are maintained in the PDF. Only one link object is created around the source phrase, but multiple destinations are stored within the link object. In order to achieve this the PDF link object has been extended to include multiple destinations, each of these destinations having an associated description allowing users to make an informed choice about which link they wish to follow. In order to use the PDF created by this plug-in, users need to have a plug-in which reads and displays the multiway links (the multiread plug-in — see later). The PDF link objects have been created in such a way that if the multiread plug-in is not installed on the end user's machine then the standard web-link plug-in should be able to read and follow the first of the link choices. Links included using the Multiway and Basic plug-ins appear to to the user as normal Acrobat web-links.
- Multiread plug-in. This is the plug-in that enables users to select a multiway link and to be presented with a choice of destinations. Users can follow the link to the destination of their choice. This plug-in also enables the user to edit either the destination, the source or the description of a multiway link. Currently this plug-in

creates an HTML file from the data in the PDF link object and uses a web browser to supply the multiple destinations to the user.

- Sendpdf plug-in. This plug-in implements the two parts of split control method 1. The user has the ability to send a PDF file maintained locally on the user's machine to a web server running a version of the Multiway OJF plug-in. As well as sending PDF, the user specifies which linkbases should be included and in what colours. The web server performs the linkbase inclusion and sends the fully linked paper back to the user. If the user is on a particularly slow link, and the paper to be used as the basis for link inclusion is available over the web, then the user can send the URL of the paper of the paper into which the server is to include the links. The server will then download the paper before including the links. The information flow for this process is described here:
  1. The user requests (by selecting a tool button in Exchange) that a web page outlining the linkbases offered by the publisher should be downloaded. (The User requires Exchange with the *sendpdf* and *weblink* plug-ins).
  2. The publisher's web server returns an HTML page outlining the linkbases offered for inclusion by the publisher. This HTML page contains a form enabling the user to specify the location of the paper that is to have the links included, the linkbases to include and the colours. (The Publisher requires a web server.)
  3. The user fills in this form and returns it to the publisher. (The user requires a web browser.)
  4. The publisher's web server requests the PDF specified by the user from a remote server. (The publisher requires a CGI script (called *inslinks*) to download the paper.)
  5. The PDF paper is sent to the publisher's web server by the remote web server.
  6. The *inslinks* CGI script then calls Exchange on the publisher's machine which automatically includes the linkbases requested by the user. (The publisher requires the *multiway* OJF plug-in.)
  7. Exchange returns the fully linked PDF document back to the publisher's web server.
  8. The publisher's web server can now return the fully linked PDF file to the user's web client.
  9. The web client reads the PDF file and invokes Acrobat Exchange so that the user can view the fully linked document.
- Utility plug-ins. These enable existing links in PDF documents to be extracted and stored in linkbases.
- Tools for generating linkbases from other information sources such as bibliography files.

Linkbase Management: Acrobat Plug-ins for a WWW Proxy

At the start of the OJF project, the existence of PDF journal articles on the WWW was conjectured rather than experienced; midway through the project most publishers had already provided archives of journal material as a commercial service, viewable not even in separate helper applications any more, but as an integral part of the WWW

browser window. In such an environment, the publishing partners were keen to see OJF software that (a) required no additional software to be downloaded and maintained on the users' system and (b) that did not change the way that the user normally browsed for information. In other words, the interactive, menu-driven link enquiry interface [2] that was inspired by Microcosm had to be changed into a hands-off, automatic interface that looked exactly like the WWW.

Hence an alternative, 'interfaceless' approach was investigated, making link addition transparent to its users by embedding it in the WWW's document transport system, compiling links into documents as they were delivered to the user by a specially adapted WWW proxy server (a proxy link server). This provides a variation on the Publisher Control model of Fig. 2: the linking proxy can be hosted by a different publisher to the one that serves the documents that the user requests—allowing the ownership of the links to be independent of the ownership of the documents.

Controlling Links In this new scenario users just see linked PDF documents delivered and displayed as normal, with no intervention on their part. Since the interfaceless link server may require at least some configuration beyond the defaults set up by the service administrator (for example choosing applicable sets of link databases for an individual), a method for communicating with the server is provided. This takes the form of a kind of a 'link remote controller' which is an HTML form displayed in a browser window and whose results are interpreted by a module in the link server and retained on the proxy's persistent storage to modify any further requests from the same user. The purpose of the controller is to give to the user the ability to choose how links are displayed and used within the processed documents

The controller gives the user the ability to choose which one of the server's installed linkbases are to be combined with requested documents, or to completely bypass the link compilation if a 'normal' document viewing mode is required. The controller provides a greater degree of control over the linking process, enabling the user to specify in some detail which link databases are switched on and off as the user browses in and out of a number of document resources, to control the kinds of linkbase that are used at such a point (e.g. internal navigation through a resource vs citation of documents external to the resource)

The Open Journal Framework Project makes use of this kind of controller to help the user navigate through large suites of collected but separate Internet resources, all integrated by the use of linkbases. By introducing a model of Internet resources (collections of documents and associated link databases) and aggregations of these resources (collections of collections of documents and associated link databases), it is possible to define the user's 'static location' in a document space, and hence to know which hypertext actions are applicable at each point in that document space. If the user travels outside all known resources (e.g. to a colleague's personal home page), then the option still remains to apply the most general links or else to have the link server refrain from applying any links. Without this model the same sets of link databases are applied to any document which the user sees.

Link Presentation Once a link is selected for inclusion in a document by virtue of its presence in a chosen linkbase and its applicability to the current document has been established (often determined by a simple keyword matching operation) the proxy inserts the link according to a specific presentation format.

The recent standard for Cascading Style Sheets for HTML documents [11] allows the presentation of many document features to be controlled by visual parameters such as font, size and colour. WWW links in HTML documents are normally tightly bound to previously marked-up anchors, and so a style-sheet's only option for parametrising link presentation is to change the typographic attributes of the (fixed) anchor. By contrast, the linking proxy has complete freedom to choose how to elaborate a link by binding it to any suitable anchor site in the document or inventing a new piece of content to act as an anchor (in the form of a distinguishing marker or a more general annotation). This freedom is balanced against the fixed layout of a PDF document which makes it difficult to fit extra visual objects within the body text. The options which apply to PDF documents are to have the links appear as boxes around the linked text (Acrobat default), footnote-style asterisks or pseudo citations. Different colours may be used to distinguish the various types of links.

Conclusions

The plug-ins outlined above have enabled hyperfeatures to be included into PDF documents after the documents have been created. The integration of Acrobat and the world-wide web has meant that hyper-features can be added remotely over the internet.

By developing tools which can incorporate external linkbases into on-line documents, end users can always be provided with up-to-date hyperfeatures. Up-to-date hyperfeatures imply not only links to pages or articles that are current (already this can be attained by the use of URN conventions) but they also enable a paper written in 1990 to link to a paper published in 1996. This ability is very difficult if hyperlinks are embedded within the context of a document (whether it be PDF, HTML or whatever). In addition, by extending the PDF link objects to cope with multiple destinations from one particular source, users have the ability to make choices about what they want to view.

Experiments with proxy-based linking allow different kinds of linking 'agent' to discover many different kinds of links [7] using a burgeoning array of supporting resource materials. In fact, one of the problems that is beginning to be encountered is how to intelligently filter the set of potential links so that the end user is not overwhelmed by a sea of blue boxes on every viewed page, representing an indiscriminate flood of links.

This work described in this paper was funded in the UK by JISC's Electronic Libraries (ELiB) programme reference 2/35. A simplified version of the linking proxy software described in the section on Linkbase Management is being commercialised under the name 'Webcosm'.

References

1. P J Brown and H Brown. Embedded or separate hypertext mark-up: is it an issue. *Electronic Publishing, Origination, Dissemination and Design*, 8(1), March 1995.

2. L Carr, D De Roure, G Hill, and W Hall. The distributed link service: a tool for publishers, authors and readers. In *Proceedings of the Fourth World Wide Web conference, Boston, MA, USA*, 1995.
3. Hugh Davis. To embed or not to embed. *Communications of the ACM*, 38(6), August 1995.
4. A M Fountain, W Hall, I Heath, and H C Davis. Microcosm: an open model for hypermedia with dynamic linking. In *Proceedings of the European Conference on Hypertext ECHT 90*. Cambridge University Press, 1990.
5. N.L. Garrett, K.E. Smith, and N. Meyrowitz. Intermedia: Issues, strategies and tactics in the design of a hypermedia system. In *Proceeding of the Conference of Computer-Supported Cooperative Work*, 1986.
6. Kaj Grénbaek and Randall Trigg. For a dexter-based hypermedia system. *Communications of the ACM*, 37(2), 1994.
7. S. Hitchcock, L. Carr, S. Harris, J. M. N. Hey, and W. Hall. Citation linking: Improving access to online journals. In *Second ACM International Conference on Digital Libraries, Philadelphia, USA*, July 1997.
8. S. Hitchcock, F. Quek, L. Carr, W. Hall, A. Witbrock, and I. Tarr. Linking everything to everything: Journal publishing myth or reality, April 1997. Presented at the ICCC/IFIP conference on Electronic Publishing T97: New Models and Opportunities, Canterbury: UK.
9. F. Halsz and M. Schwartz. The dexter hypertext reference model. In *Proceedings of the Hypertext Standartdization Workshop*, 1990.
10. Adobe Systems Incorporated. *Portable Document Format Reference Manual*. Addison Wesley, June 1993.
11. Hakon Wium Lie and Bert Bo. *Cascading Style Sheets : Designing for the Web*. Addison-Wesley, 1997. ISBN: 020141998X.
12. L Carr L, D Barron, and W Hall. Why use HyTime? *Electronic Publishing: Origination, Dissemination and Design*, 6(1), Dec 1993.
13. H Maurer and I Tomek. Some aspects of hypermedia systems and their treatment in Hyper-G. *Wirtschaftsinformatik*, 32(2), April 1990.
14. Amy Pearl. Sun's link service: A protocol for open linking. In *Hypertext '89 Proceedings*, 1989.
15. P. Smith, D. Brailsford, D. Evans, L. Harrison, S. Probets, and P. Sutton. Journal publishing with Acrobat: the CAJUN project. *Electronic Publishing: Origination, Dissemination and Design*, 6(4), December 1993.