Dynamic hyperlink inclusion in online journals

STEVE PROBETS[†]
DAVID BRAILSFORD
Electronic Publishing Research Group
Department of Computer Science,
University of Nottingham
Nottingham, NG7 2RD, UK

sgp@cs.ac.nott.uk

LES CARR
WENDY HALL
Multimedia Research Group
Department of Electronics and Computer Science
University of Southampton
Southampton SO2

lac@ecs.soton.ac.uk

SUMMARY

The two complementary *de facto* standards for the publication of electronic documents are HTML on the World Wide Web and Adobe's Acrobat viewers using PDF (Portable Document Format). A brief overview is given of these two systems followed by an analysis of why the embedded, and very concrete, nature of their hypertext links leads to great problems with keeping the 'hyperstructure' up to date.

An SGML-based syntax is presented, adapted from that used in the Microcosm system, which allows links and other hypertextual material to be kept in an abstract form in separate link bases. 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 common link-base 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, e.g at the time that the document is requested by the end-user from the publisher. These methods ensure that disseminated papers always contain up-to-date links. The techniques discussed have been implemented using a combination of Acrobat plug-ins, Web servers and Web browsers.

KEY WORDS Acrobat PDF World-wide web Automatic hyperstructure Plug-ins

1 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) {1] with its HTML markup language; the other is Adobe Acrobat where the underlying Portable Document Format (PDF) [2] 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.

World Wide Web documents are marked up in hypertext markup language (HTML)

[†] Present address: Electronic Publishing Solutions, 5 Navdeep Court, Melton Road, West Bridgford, Nottingham, UK

CCC 0894–3982/95/040001–13
Received Not Applicable
Revised Not Applicable

which is derived from SGML. Elements such as paragraphs, titles, tables and so on are tagged in a generic way and the viewers for HTML, such as those from Netscape or Microsoft, provide a generic set of seriffed, sans-serif and fixed-pitch fonts. 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 (and the underlying window system) rather than any inherent rasterising model in HTML.

Acrobat viewers, by contrast, are very definitely 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 hypertextual features in PDF including links, bookmarks and electronic ‘yellow stickers’. The Acrobat viewer software for PDF is available for all of the most popular operating systems (Windows, Macintosh, Sun UNIX etc.) and comes in two versions: the ‘Reader’ is freely distributed and allows documents (including any existing hyperstructure) to be browsed and printed out; the‘Exchange’ version has all of the Reader’s functionality coupled with facilities for editing and adding hyperstructural items and saving the amended file.

For understandable reasons publishers are adopting Acrobat whenever they want full ‘page fidelity’ with respect to the ink-on-paper version of a document. This is because, in part, the PostScript code used for typesetting a document can readily be converted to PDF by using a program called *Distiller*. Acrobat also has an extensive application programmers interface (API) [3] which enables developers to write code that can be executed under certain circumstances by Acrobat Exchange. 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 Acrobat Exchange. 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 have underlined the importance of WWW for disseminating documents whether they be in HTML or PDF. The ‘Amber’ [4] project, enables the Acrobat Reader to run in the same window as Netscape or Internet Explorer browsers. Amber exploits the plug-in capabilities of the Reader and of the chosen Web browser, to allow a closely integrated hybrid viewer to be created (rather than the unwieldy method of installing Acrobat Reader as a ‘helper-app’ for Netscape). Amber enables PDF as well as HTML documents to be downloaded over the Web, and viewed in a page-at-a-time fashion within the enhanced viewer.

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, as we go along, 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 [Intermedia, Microcosm, Hyper-G] which show that ‘separable hyperstructure’ confers enormous benefits in the flexible usage of an electronic document. By keeping hypertextual items in a separate link base we are able to engineer a publisher’s toolkit that enables hypertextual 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 our ideas.

2 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.ep.cs.nott.ac.uk/cajun.html">CAJUN</A>
project ...
```

will cause the phrase ‘The CAJUN project ...’ to be rendered on the screen, with the word ‘CAJUN’ being highlighted (usually in blue) by virtue of its position between the <A> and </A> tags. This highlighting denotes that CAJUN is to be the ‘button’ for a link and the ‘destination’, or ‘anchor’, of that link is specified by the information in the string beginning "http://.... Notice that this notation specifies very clearly *where* the destination document is held, via the Internet address www.ep.cs.nott.ac.uk (which leads to one of the Electronic Publishing Group’s servers at Nottingham). Once that server has been contacted the ultimate destination is the start of the document cajun.html. This notation has the virtue that the word ‘CAJUN’ remains the button 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 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) or inter-document (i.e. from one place in a document to a particular place in another document on the same file system). The word ‘place’ in the previous sentence has to be taken all too literally: the button and the anchor of a PDF link have to be specified as bounding boxes of the desired ‘hot areas’ on the appropriately PDF pages. A PDF version of the previous example could not tag the word CAJUN, in the abstract, as being the button of a link. Rather it assumes that, by one means or another, the position of of the word on the page, and its bounding box, are capable of being calculated so that a position-specified button can be created. To be fair, a small amount of indirection is permitted, for the destination of a link, in the latest (1.??) 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 have arguments specifying the coordinates of PDF yellow-stickers, bookmarks or link anchors. A special PostScript prolog arranges to ignore these pdfmarks when the file is printed out but if the PostScript is converted to PDF by Distiller (which is just a modified and enhanced PostScript level 2 interpreter) then the pdfmarks are mapped to the appropriate PDF hyperstructural item.

The CAJUN [5] project at the University of Nottingham has developed delayed-binding techniques and a special 'CAJUN PostScript prolog' which enables the laying-down of PDF links etc. to be automated.

More recently, a web-link 'plug-in' for version 2.0 of Acrobat enables web links. Web links store the Universal Resource Indicator (URI) address within the PDF document. When the source of the link is selected the target document is downloaded over the Internet and displayed on the client's machine. Named destinations can be used as the destination view at the termination of a web link to another PDF document. That is to say that if a PDF file contains a named destination specifying a particular view of a specific page then this destination can be the subject of a web link.

Both Acrobat and web servers are extensible, in that they provide hooks through which the functionality of the system can be extended. Web servers use Common Gateway Interface (CGI) [CGI] scripts. A CGI script is a script or program that runs on the web server with end-users supplying data to the server 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.

3 MODELS OF HYPERSTRUCTURE

The previous sections have shown that there are several interconnected issues to be addressed when developing a system for hypertextual 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 hypertextual 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 embedded in it already. Suppose, furthermore, that this file is to be viewed by 50 different people in an organisation and all of these persons 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 button 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 button 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 button at the last possible moment thereby avoiding the problems of 'hard' embedded links becoming out-of-date.

A third issue is the uni-directional point-to-point nature of links in HTML and PDF.

Separable hyperlinks are advantageous because 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 provide 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. 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 attempts to address these issues in some existing hypertext systems.

3.1 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.

3.2 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 page 7 of the file XYZ.PDF placing the coordinate (x3,y3) at the top of the visible view and making the width of the page fit the width of the window.

A world-wide web link description could be:

Link from this phrase to file XYZ.HTML and display this document with the anchor #destination at the top of the browser window.

A link from an Acrobat PDF file to a web HTML page would combine these descriptions:

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.

3.3 Microcosm

One particular hypertext system takes the idea of hyperlinks a stage further. Microcosm is an example of a recent hypertext system that [integrated information environment] utilises the idea of separable link databases known as linkbases. These linkbases contain information about what should be linked, and to what. Microcosm supports three types of links — *generic*, *local* and *specific*. A Microcosm linkbase stipulates links in terms of these types. Microcosm generic links specify that any occurrence of a particular word within any file should be linked to a particular document. A local link specifies the file in which the source word must occur and a specific link indicates which particular word in the file should be linked. The principle behind Microcosm is that given any particular file, users should be able to query their linkbases to determine which words should be made into active links. If multiple links occur from the same word then a choice of destinations is displayed to the user. The benefit of the different link types lies in the ability to scope links to particular files.

[Steve, Les. Say a bit more here. Presumably PDF links are specific? What about HTML links?]

3.4 Hyper-G

Hyper-G treats both links and documents as first-class objects, storing them in databases which are interrogated by the document viewers. Unlike Microcosm, Hyper-G links are restricted to being point to point (one-to-one unlike Microcosm's many-to-one and one-to-many links), but precisely because of this Hyper-G allows consistency checks which Microcosm does not. /** I CAN'T TALK MORE SPECIFICALLY ABOUT HYPER-G TECHNOLOGY UNTIL AFTER THE WEB CONFERENCE. SORRY. **/

```
<link type=generic><src><doc><offset><sel>Typesetting<dest><doc>
http://www.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>TeX<dest><doc>
http://www.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://www.cajun.cs.nott.ac.uk/wiley/journals/epo/pdf/volume2/
issue1/epdxb021.pdf<offset>
<sel><title>Why Use SGML?
```

4 A SYNTAX FOR ABSTRACT HYPERLINKS A link database (as shown in Figure 2) is described by a simple SGML DTD. 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.

A link is of type specific if its source anchor is constructed from a complete triple (i.e. a specific occurrence of a selected object in the named document), local if its anchor is ignores the offset component of the triple (i.e. any occurrence of the selected object in the named document) or generic if only the selection is used (i.e. any occurrence of the selected object anywhere in any document). Note that this flexibility in specifying the source anchor means that a single link to a destination may appear in many places at once, giving rise to a number of useful features for the hypertext author.

Further fields in the link database allow a title to be specified, as well as the author of the link and the time that it was created. The timestamp is particularly useful in checking any potential problems with documents that have been edited after being joined by a link.

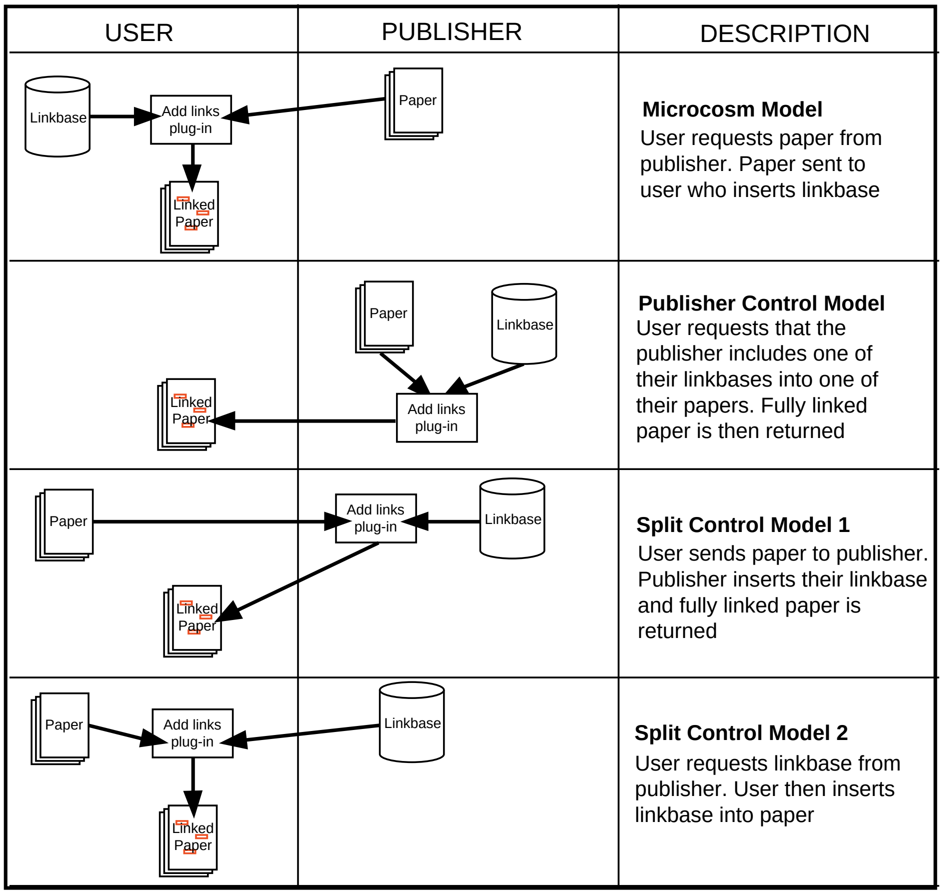
Figure 1. An example Linkbase

5 FOUR MODELS FOR DISTRIBUTED LINK SERVICES

The idea of external linkbases (for an example of a linkbase see figure 1) is attractive in the current climate.

The rapid expansion of available data on the web 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 as information has to be updated in one place rather than in multiple files.

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-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 places and depend on who owns the papers and the linkbases. These models are described below and outlined in figure 2.



Microcosm Model
User requests paper from publisher. Paper sent to user who inserts linkbase

Publisher Control Model
User requests that the publisher includes one of their linkbases into one of their papers. Fully linked paper is then returned

Split Control Model 1
User sends paper to publisher. Publisher inserts their linkbase and fully linked paper is returned

Split Control Model 2
User requests linkbase from publisher. User then inserts linkbase into paper

Figure 2. Four Models of Link Inclusion

5.1 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.

5.2 Publisher control model

Another model is 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.

5.3 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 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 interst then links from the fish biology linkbase could be inserted in one colour and links from the radioactive isotope linkbase in another colour.

5.4 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.

6 THE ACROBAT PLUG IN TOOLKIT

The Open Journal Project is a project undertaken by the Universities of Southampton and Nottingham. 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 is to develop tools which enable the second of the twsplit control o 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 be becoming the standard format for journal article dissemination, 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:

- 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, then all these links are maintained in the PDF. 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.
- 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. A diagram outlining the information flow in the latter case can be seen as figure 3. The web server uses a CGI script to obtain the PDF, read the user requests and run Exchange with the multiway OJF plug-in to include the linkbase. The numeric stages in figure 3 are:

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

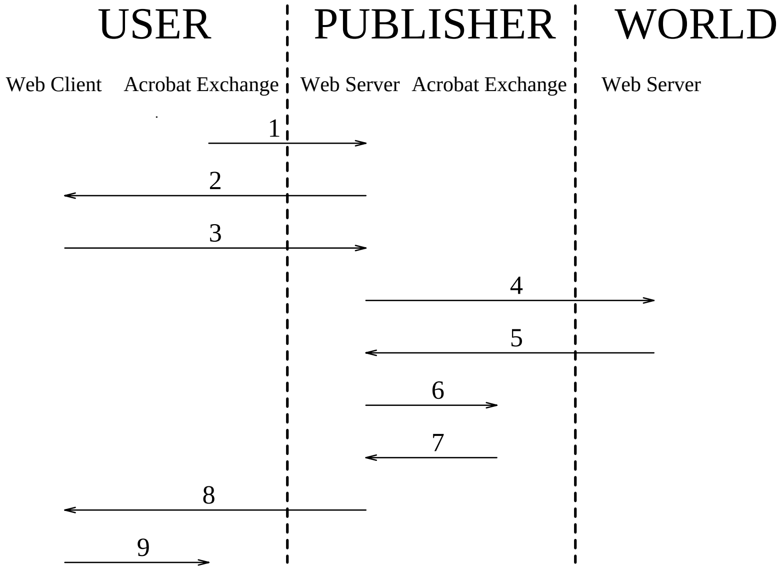


Figure 3. Information Flow in the Sendpdf Plugin

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 Exchance so that the user can view the fullyliked 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 LATEX bibliography files.

6.1 Linkbase maintainance and intelligent agents

Currently the creation of linkbases is essentially a manual process, although tools have been developed to creat linkbases from LATEX bibliography files. The need to maintain up-to-date linkbases is vital and this is one area where there is an ever increasing need for intelligent agents. Intelligent agents are programs which have an intrinsic knowledge of the Web. Agents would be able to keep track of, or find out about, the whereabouts of papers and make sure that users are supplied with these resources from the most sensible server. Agents could not only track papers but could also contain information on bibliography listings, linkbases etc. Tools for linkbase creation could be adapted to use intelligent agents to their fullest extent. Linkbases conform to a simple SGML DTD (see figure 1) and therefore linkbase maintenance via intelligent agents is a viable proposition.

7 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 not only means links to pages or articles that are current (as this can be attained by the use of URN conventions) but it also enables 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.

REFERENCES

1. T. Berners-Lee, R. Cailliau, and J. Groff, ‘The World Wide Web’, *Computer Networks and ISDN Systems*, **24**(45), 454–459 ( ).
2. Adobe Systems Incorporated, *Portable Document Format Reference Manual*, Addison-Wesley, Reading, Massachusetts, June 1993.
3. Adobe Developer Support, ‘Acrobat Viewer Plug-In API On-line Reference’, Technical Note #5168, Adobe Systems Incorporated (15 September 1995). Only available with Acrobat SDK.
4. D.W. Barron and Netscape Inc., ‘Ye Olde Amber Plugge-Inne’, *Electronic Publishing — Origination, Dissemination and Design*, **2**(1), ( ).
5. Philip N. Smith, David F. Brailsford, David R. Evans, Leon Harrison, Steve G. Probets, and Peter E. Sutton, ‘Journal Publishing with Acrobat: the CAJUN project’, *Electronic Publishing — Origination, Dissemination and Design*, **6**(4), 481–493 (1993).