Lessons from the First Hypertext Digital Library, The NLS/Augment Journal System, A Hypertext Management System (HMS)

Raymond R. Panko
University of Hawaii

Abstract

Doug Engelbart's NLS system introduced working hypertext in the early 1960s. The NLS link had a syntax similar to today's HTTP uniform resource locators (URLs). NLS links pointed to specific files in specific directories on specific machines. This created a loose hyperbase, that is, an unmanaged collection of linked documents. In the late 1960s and early 1970s, Engelbart's team at Stanford Research Institute added a formal hypertext digital library, the Journal system. The Journal system added a management layer to the hyperbase. The Journal system used a new approach to locating documents. Instead of asking for a document by location, you asked for it by accession number. The Journal system managed the physical retrieval from disk or archive tape. This paper discusses the Journal system and suggests how its managed approach to retrieval can be extended to provide richer tools for document access. In this expanded approach, access will be made through a hyperbase management system (HMS), which will not only hide the document's location but will also provide new search tools and new types of links among managed documents described in the HMS's document information base (DIB). This approach would require a new type of URL, the managed URL.

Introduction

In the 1960s and 1970s, Doug Engelbart's team at Stanford Research Institute developed the *NLS* system, later renamed *Augment*. NLS/Augment introduced many innovations, including the mouse and hypertext.

Initially, NLS/Augment focused on augmenting individual knowledge workers [Engelbart, 1962]. Gradually, however, the focus began to shift to group support. In many ways, NLS/Augment became the first working system for office automation, groupware, and computer-supported cooperative work.

One of the fruits of this focus on teamwork was the creation of the Journal system [Engelbart, 1984]. The Journal system was the world's first *managed hyperbase*, which we will define as an organized collection of hypertext documents. Previously, the NLS hyperbase had been unmanaged. Only the links among documents created any organization. The Journal system added a management layer to the hyperbase.

This paper describes linking in the early versions of NLS. It then describes the Journal system, including its new approach to organizing documents in a digital library and its new form of link. The paper then describes how the approach used in NLS can be generalized to provide more functionality. This generalization will require a new type of link, the managed link.

Original Links in NLS/Augment

Even the earliest versions of NLS used hypertext. The author of a document could add links to a document. The link was delimited by left and right angle brackets. Its syntax (using simplified terminology) was roughly <*host,directory,file,location*> [Engelbart, 1984]. "Location" was a location within a file.

This syntax is obviously similar to the World Wide Web URL syntax defined in RFC 1738 by Berners-Lee, Masinter, and McCahil [1994], which also identifies a specific file in a specific directory of a specific host computer.[1] However the NLS link syntax was actually somewhat more sophisticated. For instance, for the file, you can use *filename.n,* where *n* was an integer. If *n* was two, then the jump would take you to the named file, then find the second link, and finally jump to where that link leads. This was called an indirect link.

For the in-file location, in turn, you could type something like *Refs "Brown"*. This would take you to the statement named *Refs* (perhaps the beginning of the references section), then to the first statement containing the text string *Brown*.

In fact, the location field had a very different purpose from a URL's optional fragment ID segment, which is separated from the path by a hash sign (#). The URL fragment identifier is not even sent by the browser to the webserver [Berners-Lee, 1994]. Rather, it is retained within by the browser and is used for showing the correct part of a document once the document is delivered.

Journal System

NLS/Augment evolved over time to support groupwork [Engelbart, 1984] (specifically the development of NLS/Augment). One obvious need was the creation of a digital library for reports and memos documenting the development of the system.

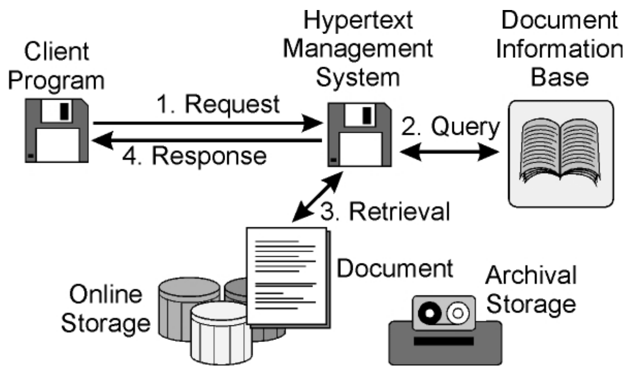
For this purpose, the group created the Journal system. The Journal system had an electronic mail subsystem. Although members of the group typically used Internet (then ARPANET) electronic mail for small messages, they usually turned to Journal Mail for longer documents. NLS was a full-featured mail system. However its most important feature for our purpose is that it asked the sender if he or she wished to record the document [Engelbart, 1984]. If the user decided to record the document, two things happened. First, the Journal system assigned the document a serial accession number. Second, receivers did not receive the entire file like a modern attachment. Instead they got a short message with an abstract and a link to the document.

Note that not all transmissions were recorded. Users only recorded those that they judged to be of permanent or at least persistent value. For the World Wide Web, Berners-Lee [1994] has reserved the term universal resource name (URN) for designations to documents designed to be persistent.

The link in NLS Journal messages link had a different syntax than the original NLS link [Engelbart, 1984]. Its syntax was <*journal, accession-number,location*>. Here, *journal* is a named journal containing recorded hypertext documents. At Stanford Research Institute's Augmentation Research Center, the journal was named Augment, so if you wished to access item 3334 and did not care about in-document location, your link would be <*Augment,3334,*>. Augment was never intended to be the only journal, and indeed, some other journals were built.

Note how different this is compared to the basic link. Instead of accessing a host, you access a journal. Instead of accessing a specific file in a specific directory, furthermore, you just give the accession number. The file might be in any directory or might even have been archived to tape. It is up to the journal to find it for you.

Figure 1: Journal System Link Retrieval



Conceptually, as shown in Figure 1, your request message went to a *hyberbase management system (HMS).* Based on the accession number keyword, the HMS located the document on disk or noted that the document was archived. If the document was archived, you would be asked if you wished it dearchived. If you typed "Yes," the HMS would send a message to the operator asking that the

[1] Actually, RFC 1630 [Berners-Lee, 1994] notes that the path should *not* be viewed in terms of specific directory path on a hard disk followed by a file name. In practice, however, this is almost universally the case for webpages.

tape be mounted. Later, when you tried the link again, you would find it.

Note that using such a link is more abstract than a traditional NLS/Augment link or a modern World Wide Web URL. A traditional link is an *explicit link.* It points to a particular file in a particular directory. One major problem with this that if a file is deleted, archived, or moved, you receive a single uninformative error message (in the case of the 'Web, the famous 404 error message), which merely tells you that the webserver software cannot find the specified file in the specified directory.

In contrast, Journal system links were *managed links.* They merely pointed you to the proper journal and passed the accession number to the journal. The Journal system did the rest of the work needed to find the information. Although the NLS Journal system was largely limited to this lookup function, we will argue below that the HMS approach can be expanded to provide a much richer set of responses.

Problems with the Journal System

Although the Journal system worked fairly well, it was not without problems. For one thing, the system did not provide an easy way to find accession numbers for postings. You had to manually wade through your old electronic mail to find the desired link. In 1974, the author sampled 500 Augment journal files. Only 24% contained any link to another document. Only 11% had more than one link. Interestingly, 11% of the documents were not sent to anyone; they were merely stored "for the record."

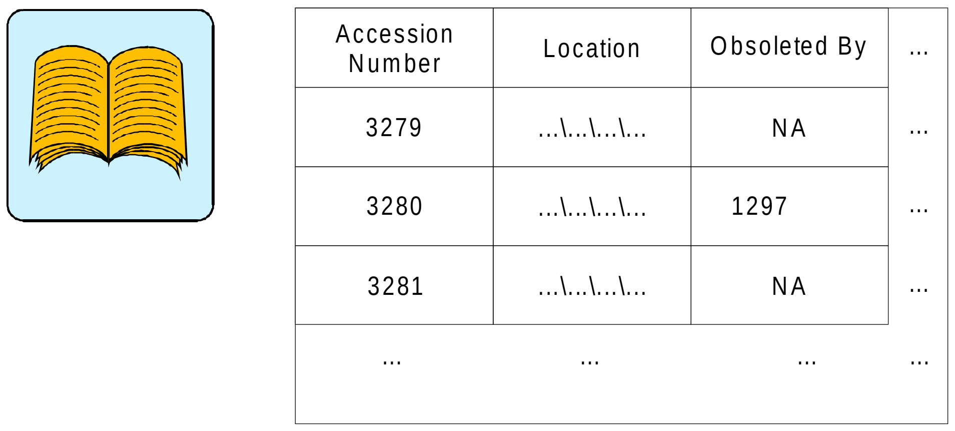
More importantly, when the Journal system was developed, it was envisioned that there would be rich information stored in each document. For example, one type of link would allow the author of a document to note that this document renders and older document obsolete. The link would give the accession number of the older file.

The most useful new types of links to be stored in documents, however, were *back links*, which could point in the opposite direction. For example, if you reached an obsolete document, there would be a link pointing to a newer document.

Unfortunately, back links were not implemented. First, a major issue was that once a document was posted to a Journal it should not be changed [Engelbart, 1984]. Adding a link, even for information purposes, would violate that principle.

Second, the process should have been automatic. In other words, if you added a link to one document saying that it obsoletes another, then a back link should have been added automatically to the earlier document, pointing to this new document. Lack of programming resources prevented this from being done. In addition, the earlier obsolete document would almost certainly have been archived to tape (this was done automatically after 30 days of nonuse), and this would complicate the problems.

Figure 2: Document Information Base (DIB) Containing Metainformation



Accession Number	Location	Obsoleted By	...
3279	...\...\...\...	NA	...
3280	...\...\...\...	1297	...
3281	...\...\...\...	NA	...
...	...	...	...

Expanding the Document Information Base (DIB)

We agree that documents should never be modified after posting. This could lead to serious problems in organizational contexts. However we also believe that modifications are not necessary for back links and similar information.

As shown in Figure 2, we argue that metainformation containing such information as relationships among documents should not be stored as links within documents. Rather, metainformation should be stored in a *document information base (DIB)* associated with the hypertext management system (HMS). This is a profoundly different view of links.

The NLS/Augment Journal system was a step in the right direction, but it was too small a step. Its DIB merely stored the location of the document on disk or on a backup tape. It would not be a major step, however, either conceptually or practically, to expand the DIB to hold other metainformation.

For example, suppose that a new document renders an older document obsolete. It would not be difficult for such a declaration to update both the new and obsoleted document entries in the DIB. In the newer document, the *obsoletes* field would be given the accession number of the older document. In the older document, the *obsoleted by* field would receive the accession number of the new document. Subsequently, if someone requested the older documents, the lookup process would note the value in the obsoleted by field. The user would then get a notice that the document was obsolete and would be given the option of retrieving the newer version.

In other words, the Journal system had the right basic technology. It simply failed to take fully into account the possibility of storing metainformation in the HMS's document information base rather than in the documents itself.

Managed URLs

The current HyperText Transfer Protocol defines URLs as explicit links. The format is shown below for HTTP.[2] This *explicit URL* points to a specific location in a specific file in a specific directory of a specific machine. We have already pointed out that this is not good if a document is deleted, moved, or archived. In addition, there is no way to know anything about the document other than its location.

http://host/directory-path/filename.ext#location

We suggest that we need to have a *managed URL* that would point not to a physical location but to a hypertext management system responsible for the file. This would lead us to something like the following:

httpm://hypertext-management-system-name/accession-number#location-in-file

Here, the link points to a hypertext management system by name. There is no specification of a host computer name.[3] This would require something like the Domain Name System (DNS) to associate HMS names with specific host computers. Another approach would be to create managed http (httpm) and use the following URL, which points to a hypertext management system on a particular computer.

httpm://host/hypertext-management-system-name/accession-number#location-in-file

One final note is that including a location-in-file fragment after the hash mark (#) is controversial. In URLs, as noted earlier, fragment ID is not even sent to the server and is only a matter for the browser to consider after the document is delivered [Berners-Lee, 1994]. In fact, the fragment ID is useless in living documents that change over

[2] Other access services, such as FTP, have different URLs. In general, a URL begins with the name of a registered access method, followed by a colon, and then followed by a path. The path syntax is different for each access method.

[3] There is precedent for this. The news URL syntax used to access USENET newsgroups specifies newsgroup names or news item ideas but not a specific USENET host. In this case, however, all USENET hosts should contain almost the same information; in contrast, different journals would contain different information. So not specifying a specific host causes no problems; any USENET host that carries the right newsgroups will do. In a similar vein, people cite RFCs generically, assuming that the reader will know that there is an RFC name space and that multiple docubases give you access to these RFCs. However, while there are multiple HMSs for RFC documents, they follow no generic HMS/DIB standard.

time [Berners-Lee, 1994]. While this is true, it is also true that you sometimes only wish to retrieve some parts of a document, for instance, certain records in a database document or the glossary of a book.

Hypertext Management Systems and URIs

Earlier in the paper, we characterized URLs as consisting of protocols, host names, directory names, and file names. This was an injustice to Berners-Lee's [1994] vision. Although we noted this in a footnote, it is time to treat the matter more fairly.

Actually, Berners-Lee viewed Uniform Resource Identifiers (a broader term) as having two parts. One was the scheme. The other was the scheme-specific part. So if HTTP is a scheme, the remainder of the URL should not be viewed as the host, directory name, and file name.

Of course in most cases, URLs are *precisely* protocol (scheme) names followed by the host name, the directory name, and the file name. However this should not blind us to broader vision possible in the URI vision. In fact, our managed URL approach can be viewed simply as a scheme or class of schemes.

In the Internet community, there currently are a number of efforts to create schemes that allow many of the suggestions we have included here. The most ambitious approaches revolve around uniform resource names (URNs) which are precisely for resources that are intended to be permanent or at least of long duration. Unfortunately, it is beyond the scope of this paper to review all of these efforts and compare them on a point-by-point basis with our own.

Conclusion and Directions

We argue that the NLS/Augment Journal system was a major step in the right direction, replacing explicit links with managed links pointing to a hypertext management system (HMS) rather than directly to files. By adding a level of indirection, the Journal system opened the door to great flexibility and functionality.

Unfortunately, the Journal system took only too a small step in the direction of storing information about the document in a document information base (DIB). It only stored the item's location on disk or on an archived tape. It did not store rich information about the document that would still be relevant even if the file were deleted, moved, or archived.

To have managed URLs, which would point to HMSs instead of to physical file locations, it would be best to have a tool like the current Domain Name System (DNS). When the browser was given a managed link, it would look up the specific host on which the target HMS was located. Otherwise, there would be a need for the author of the link to know the IP name of the HMS host. This would also make it more difficult to move the HMS to another host computer if there were a need to do so, because links would point to the old host computer.

Moving toward HMSs with rich document information bases (DIBs) would have another benefit, namely searching. Suppose that a HMS manages a university department's webpages. It could be possible to do searches on the document information base (DIB). For instance, we may not know the accession number of a particular number, but we might know something about it, such as its title or author. The HMS could use this information to search for a paper within the DIB, giving the accession numbers of matches. More generally, searching would allow you to see all papers by a certain faculty member or on a certain topic. In general, it would be possible to store many types of metainfirmation about single webpages, webs of related documents, and entire DIBs. It would radically extend the capabilities of today's World Wide Web.

References

Berners-Lee, T., *Universal Resource Identifiers within WWW*, RFC 1630, June 1994.

Berners-Lee, T.; Masinter, L.; and McCahill, M., *Uniform Resource Locators*, RFC 1738, December, 1994.

Engelbart, Douglas C., *Augmenting Human Intellect: A Conceptual Framework,* Summary Report, SRI Project No. 3578, Stanford Research Institute (now SRI International), Menlo Park, CA, October 1962.

Engelbart, Douglas C., "Authorship Provisions in Augment," *IEEE Compcon Conference*, 1984, reprinted in Irene Grief, *Computer-Supported Cooperative Work: A Book of Readings*, Morgan-Kaufman, San Mateo, California, 1988, pp. 107-126.