Surfing the Web Backwards

Soumen Chakrabarti* David A. Gibson† Kevin S. McCurley‡

December 14, 1998

Abstract

From a user's perspective, hypertext links on the web form a directed graph between distinct information sources. We investigate the effects of discovering "backlinks" from web resources, namely links pointing *to* the resource. We describe tools for backlink navigation on both the client and server side, using an applet for the client and a module for the Apache web server. We also discuss possible extensions to the HTTP protocol to facilitate the collection and navigation of backlink information in the world wide web.

1 Introduction

Hypertext predecessors to the world wide web were conceived as bidirectional in nature, requiring the authors of both resources to create links between the two. Such a system promotes a uniform and consistent set of hyperlinked documents, but inhibits the modern scientific tradition of open criticism and debate. A large part of the intellectual value of the web is derived from the fact that authors of web documents can freely link to other documents in the namespace without agreement from the author of the referenced document. Placing a document into the namespace implicitly implies an ability to link to that document, and this is part of the reason that the web has grown so explosively.

Scientific literature has traditionally built on the body of knowledge that preceded it, and the notion of citation has become a fundamental part of scientific writing. By following citations, the reader is able to trace backward in time through the evolution of ideas that leads to the current work. Unfortunately, this system is unidirectional, and does not allow the reader to trace forward in time. Such a facility is provided as a commercial service by the Scientific Citation Index[1] and the Web of Science[2] from Information Sciences Institute. Using the Science Citation Index, a reader can follow citations of earlier scientific work, and move forward in time through the scientific literature.

In the context of the world wide web, we call such a reverse citation a *backlink*. Backlinks are seldom used in the world wide web, but we believe that they add significant value to the process of information discovery. Through the use of our tools we have discovered many unusual "nugget" sites that are not easily discovered by tracing forward along the edges of the web graph, but are quickly located by traversing the links in the reverse direction. Following backlinks generally leads to information that has been created more recently, and may therefore provide an important

*IBM Almaden, soumen [at] almaden.ibm.com

†U.C. Berkeley, dag [at] cs.berkeley.edu. The work was done in IBM Almaden.

‡IBM Almaden, mccurley [at] almaden.ibm.com

[1] `http://www.isinet.com/prodserv/citation/citsci.html`

[2] `http://www.isinet.com/prodserv/citation/websci.html`

improvement to the timeliness of information discovered on the web. As the web matures over time, we expect this factor to increase in importance.

Backlinks enjoy an advantage over automated "find similar resources" mechanisms because they are human-created. Other techniques, such as HITS [9], have already exploited this fact to great success. Page creation is by and large a personal authoring process, and creating links involves quite some deliberate effort and judgement, certainly far more than any automated system can dream of doing for the foreseeable future. The reason for creating the link might vary from comment to compilation to criticism, but if the author has some standard of quality, there is always some relevancy.

Both HITS and backlink browsing are greatly aided by *hub resources*, which are compiled lists of relevant links on particular topics. If a site is respected enough to appear on one such resource list, one can find a list of similar resources with a single backlink traversal. This is probably the most useful backlink browsing technique. If backlink browsing became more commonplace, we would expect that commentary and criticism resources would become much more effective and numerous as well.

The effort required to maintain simple backlinks is minimal. Search engines provide a reasonably effective backlink database already. As discussed in the remainder of this paper, server maintainers can provide a more complete and up-to-date database very easily, and greatly enhance the usefulness of their sites. In effect, a resource can increase in quality without its creator's intervention, by the efforts of people authoring related resources.

This combination of forces leads us to believe that a significant enhancement of the world wide web can be achieved through backlink navigation. We propose simple extensions to the HTTP protocol to facilitate navigation through backlinks. We have implemented these facilities in a Java applet to augment existing clients, and have also built a module to support backlink retrieval from the popular Apache web server. These are not the only tools that one might imagine to facilitate backlink navigation, but provide the basis for building a cooperative infrastructure for open dialog. In section 3 we describe our simple extensions to the HTTP protocol. In section 4 we describe the design of the Apache module for the Apache web server.

1.1 Discovering Backlinks

There are already a few methods of discovering backlink information on the web, and we intend to build on these. Clients and servers can already consult various search engines (e.g., [8]) for backlink information, and we expect this to grow in popularity. In addition, servers can track backlinks from their resources by the Referer[3] request-header of the HTTP protocol [7, section 10.13]. Thus, servers could supply this information to clients interested in it. Backlink information is already available through several search engines, supported by advertising on those sites. We expect that this will continue for some time to come, although it does not scale well as a generic facility for backlink navigation if it is invoked on every client document retrieval. From a client's perspective, they might prefer the independence of a search engine for supplying backlinks. However, crawling has some other drawbacks as well, notably that there is a time delay from the time that a link is created until the time that the crawler returns to crawl the link. This may significantly degrade the value of backlink information for researching time-sensitive information. Moreover, search engines are primarily dependent on crawling the web from known sites, and cannot discover sites that have no links into them.

[3]Note that the correct spelling of the word should be Referrer, but for some reason the alternate spelling is used in the HTTP specification.

A more scalable solution for both clients and servers is to leverage the existence of the Referer request-header. If servers recorded backlink information from the Referer request-header, then clients could retrieve backlink information directly from the server of the target resource. This has the advantage that it decentralizes and balances the load of backlink browsing, and leverages the existing practice of providing Referer request-header information.

1.2 Barriers to Acceptance

Building an infrastructure to support retrieval of backlink information will require a certain degree of cooperation on the web. Relying on servers to collect and redistribute information derived from the Referer field provides the most scalable solution, but it requires the cooperation of the server being linked to. The scientific community generally endorses open commentary, so it is natural to assume that this community would embrace the ability to find out who makes reference to published work. Sites and authors with an appreciation for freedom of expression and peer review can be expected to embrace the notion of backlinks, but those more interested in "controlling the message" may choose not to.

For example, many commercial sites are unlikely to provide unfiltered backlink information, since they often have a different motivation in their publication, and are more strongly interested in controlling the message. Government sites may also be hostile to the idea of divulging information that criticizes government policies expressed on their web pages. Moreover, the ability to freely link to a document is not universally accepted, and some sites filter access to resources on the basis of the Referer header. There is no mechanism that can compel an author to acknowledge opinions or citations that they disagree with, and we see no conflict in this. We do not advocate any form of coerced compliance for building a backlink navigation capabilities, but expect it to emerge through a combination of balancing forces in society.

The reason for our optimism is that, while some sites may consider it counter to their interests to supply backlinks to their resources, other sites may leap at the opportunity. For example, the League of Women Voters considers it their mission to encourage citizen participation in the government process, and may therefore offer a "portal site" providing backlinks to government sites. Consumer's Union (publisher of Consumer Reports magazine) offers noncommercial product information for consumers, and backlink information for commercial sites might well be viewed as providing consumer information. Such a service might even be purchased by the client or supported by advertising.

Since a resource's backlinks provide a public "place" for people to comment, backlinks can be abused like many other public information channels. If a simple-minded approach is used to compile backlink information, then it becomes vulnerable to a form of "spamming", where a target site is induced to show a backlink to another site. In the case of backlinks, there are a variety of deterrence measures that can easily inhibit such attacks without requiring human management of the backlink mechanism. This is related to the problem of how to manage a backlink infrastructure, and we will return to the topic in section 4.

1.3 Privacy Concerns

The Referer header of the HTTP protocol has always been identified as a potential threat to privacy, since it conveys information about how a reader found some information. In addition, servers that receive the referer information are supplied with a tool to discover other resources that link to them. One of us (KM) has for several years maintained a satirical site called DigiCrime[4] that

[4] `http://www.digicrime.com`

parodies Internet-related hacking and criminal activities. The referer entries from the log files have regularly proved to be a supply of hacker site URLs that belong to a "web underground" of sites that supply information on computer security from a hacking point of view. These sites often exist only for a short time, because they provide access to hacking tools. The creators typically only share the existence of such sites with a few friends, and the sites sometimes exist only in isolated subgraphs of the entire web. By mining the log files of a site linked from these isolated resources, their existence can be exposed.

Intranets isolated by firewalls are also vulnerable to leaks through the Referer field. For example, at one time there existed a link from inside the FBI firewall to DigiCrime, and this fact was discovered by mining the log files (the author was unable to access the page). If there was an internal Sun web page with the URL `w3.sun.com/javaos/parallel/99plan.htm`, and if this resource linked to a competitor web site, then the mere transferrence of the Referer field in the HTTP protocol may leak the existence of a plan for deploying a parallel version of JavaOS in 1999 (there is no such plan that we are aware of). By making backlink information publicly available, our extensions may accelerate the leakage of such information. Luckily there is a simple solution for corporate intranets - namely to configure firewall proxy servers to remove or modify any Referer headers from internal sites. Another alternative is to use only browsers that follow the recommendation of RFC 2068 [12] to allow the user to control when Referer information is transmitted, in much the same way that the user is commonly allowed to control cookies. This recommendation is currently ignored by most popular browsers.

On the public web, the privacy impact of sending the Referer field is limited to those sites that are not linked from other resources. Creators of documents may expect their resource to remain private because they do not advertise it or provide links to it from other public resources. Such "security by obscurity" will be further degraded by the propagation of backlinks, but access control measures already exist to counter this.

1.4 Metadata efforts

Hypertext has been around since the Talmud and the Ramayana, if not earlier. In the modern world, as early as 1994, Nelson and Walker proposed Xanadu, a hypertext database that supported annotated bidirectional hyperlinks [21]. In a prototype called Zigzag, documents and links were rendered, edited, and navigated using Perl and the Curses library. In 1995, the early days of the web, Walker proposed Hack Links: a collection of CGI programs that gives some of the link annotation functions without any need to change the document server [26]. References on hypertext systems can be found in [5, 10]. More recently, the Foresight Institute has revived interest in the use of backlinks in hypertext publishing systems [15].

We can only speculate why these architectures have not become immensely popular on the web. Most likely, they were far ahead of their time, long before even the early days of standardizing, implementing and extending web protocols and markup languages. Storage costs may have been a barrier, or perhaps the small size and narrow interest in the web made it easier to locate resources at that time.

Today, the classification and organization of information on the web is a major problem for researchers seeking out specific information. Several researchers have previously suggested that an increased use of metadata will help in this direction, but retrofitting this information to the existing web is very challenging. Backlinks from a resource can be viewed as a form of dynamic metadata about the resource. There are several mechanisms that either exist already or are in the planning stages to support the use of metadata for web resources. These include the evolving HTML specification, the Resource Description Framework (RDF) [3], XML, WebDAV [13], and the

Dublin Core Metadata Initiative [6].

Backlinks can also be used to bootstrap the effectiveness of other metadata. One of the problems identified by Marchiori [18] was the back-propagation of metadata, in which metadata from two objects may be merged if one provides a link to the other. A significant proportion of the existing web will lack metadata for the foreseeable future, but the ready availability of backlink information could be used to fill in these gaps across resources that don't incorporate metadata.

1.4.1 Metadata in HTML

HTML 4.0 [2] contains various ways to embed metadata into it, providing information about both forward and backward links. This can be used to convey backlink information to clients, but can also be used by other backlink information sources to improve the quality of information available about a link. If a server discovered a backlink via the HTTP Referer header, then it may choose to crawl the source of the link to retrieve information about the link (such as a document title for the resource). Though few resources support it yet, version 3.2 of HTML introduced a relationship attribute for links, which later became LinkType in HTML 4.0. This allows HTML links to have a purpose attribute such as "copyright" or "glossary". Links may also have a title attribute as well as content that describes the link.

As a means of conveying backlink information to clients, HTML is deficient for several reasons. The HTML `LINK` element describes a relationship between the current document and another document. Unfortunately, there was no LinkType defined for "is referenced by", and the closest one in the DTD is "prev", for use in an ordered set of documents. We could easily extend this to include a "back" LinkType indicating knowledge of a link from that document to the current document. HTML 4.0 also specifies a way to insert META tags into the head of a document, and this could be used to store and transmit backlink information. There is no formal specification of how these should appear in documents, and documents have tended to implemented them in an ad-hoc manner that is consistent only within an organization.

Probably the biggest obstacle to conveying backlink information through the destination document is the fact that most of the documents that currently exist in the world wide web consist of static documents residing in file systems, and mere insertion of metadata is not enough. Much of the metadata (in particular, backlinks) is dynamic and requires a management infrastructure to update it. Moreover, many (if not most) of the existing resources have been abandoned by their authors, but the information retains value. Updating these resources will require a signficant effort to retrofit them with appropriate metadata.

Finally, it should be pointed out that a significant amount of information on the world wide web is contained in other data formats besides HTML. Some of these data formats now support external hypertext links. Examples include XML, PDF, FrameMaker, WinHelp, and Lotus Notes. Extracting consistent metadata from all of these sources will prove problematic. We therefore believe that the predominant method for transporting metadata in the near term will be external to the retrieved resource. We will return to this in section 3.

1.4.2 RDF and WebDAV

Work is underway within the W3C to design a Resource Description Framework [3] to express metadata. RDF is an application of XML, and may ultimately provide a rich framework in which to express arbitrary metadata, including backlinks. For example, the Dublin Core includes a relation type of `IsReferencedBy` that is evidently intended to express backlink information. Unfortunately, this effort is in its infancy, and as of yet there is no specification for how to express schema for

RDF.

RDF does not address the problems of discovery and retrieval of backlink metadata. Retrieval is addressed in part by WebDAV and DASL, but discovery from multiple sources seems to be mostly overlooked. WebDAV supports retrieval of metadata as a subset of a much more ambitious effort to support distributed authorship of web resources. The design goals of WebDAV (see [24]) include support for locking, versioning, collections, remote editing, and namespace manipulation. The current draft includes new HTTP methods called PROPFIND and PROPPATCH to support retrieval and update of metadata about a resource. There are several potential barriers to rapid adoption of the WebDAV protocol, including potential security risks for servers.

Another thing that is not addressed by WebDAV is the ability to make queries on metadata. This is addressed in the DAV Searching and Locating (DASL) [4] protocol, which is a proposed extension to HTTP using XML. Among other things, this allows the specification of simple queries on DAV data. We expect WebDAV and DASL to play a significant role in the manipulation of backlink metadata.

1.5 Tools and Goals

Our tools focus on metadata that can be automatically generated from the existing web infrastructure, but this may not fulfill the needs of all parties. Backlink metadata may be constructed from a variety of sources, including human editing, crawling, and automatic complilation. Servers may also consult independent web crawlers to discover links that are infrequently traversed. Servers may also store records of link traversals to be used in ranking and reporting the use of information. Backlink data can also be ranked and filtered by a variety of strategies. Clients may retrieve backlink information from a variety of sources, including the server holding the destination resource or various third parties. More sophisticated tools would be required to leverage these various sources. Referer databases and crawlers are still a valuable resource because they can provide a data source for site maintainers and browsers.

It is not our goal to address the general design of a management system for backlink metadata, primarily because the specific goals behind metadata vary according to whose interests are being served. Users may wish to consult multiple sources of backlink information, and may wish to promote the free exchange of link information. Authors may not wish to provide pointers to things they do not agree with. Sites that wish to control the quality of backlink information more tightly, or exercise some editorial control on backlink information may choose to institute a labor-intensive human editing process on metadata (although backlink information could also be incorporated into the resources themselves). Third parties may wish to supply backlinks for their own purposes about resources they do not own. Supporting all of these will require a diverse set of tools. Our goal is simply to demonstrate that backlinks have value for information discovery, and describe the design of some basic tools to exploit them.

2 Client design

One of the primary reasons for the widespread success of hypertext beyond FTP and Gopher is the availability of an effective graphical user interface for users to navigate through the web. In order for backlinks to become a useful augmentation to the existing world wide web, there will have to be a widely deployed integration of the information into the user interface.

We are certainly not the first to have suggested displaying information related to the graph structure of the web within the browser. Many programs called *site mappers* have been developed that perform site-level crawls to help administrators maintain links. One example is Mapuccino

from IBM Haifa [16]. These have been generalized and made more sophisticated. Miller and Bharat [19] have designed a system called SPHINX which is a customizable crawler which analyzes and presents sections of websites in a graphical format.

In this section we describe our experiences in implementing the "browse assistant" applet. We currently have a version working with Netscape Communicator[5] version 4.07 or 4.5. We wrote the applet to study the effect of backlink guidance on the browsing activity and experience, and as a proof-of-concept prototype of the client end of our architecture proposal. Currently the browse assistant consults a search engine such as HotBot[6] to retrieve backlink information. Views of the browser assistant are presented in Figures 1–4.

2.1 User interface goals

Due to the wide variety of applications and cultures present on the web, it is a difficult task to cater to all users in the transparently simple way that forward hyperlinking does. This makes the client design particularly tricky. The information has to be presented in as useful a way as possible, while being as universal as possible, and of course staying within the realm of feasible technology.

For this reason we designed the client in as general a fashion as possible, to make it useful in the widest variety of contexts. Of course, task-specific additions to this basic framework (such as finding related news stories, or having more sophisticated navigation controls) would improve it in specific contexts. But it has been our experience that a generic display has considerable use on its own.

The guiding principle in the client design is to provide *context* for the user browsing the web. Studies have shown that browsing the web is most strongly visualized with spatial metaphors [17], and context is best thought of in these terms: your context is "where you are" and where you are able to go next.

Web browsing is seldom a simple forward progression of link clicks: one often backtracks and explores new paths. However, the current generation of browsers show only the current page, and its outgoing links, as context. Our client expands this context in a natural way by making it easy to access pages you have visited recently, and by providing the backlink information.

Thus the two main interface panels are a history view, which shows recent browsing context, and the backlink view, which lists the backlinks of the currently displayed "target" page, showing the context of the target page in the larger compass of the web. The history view is displayed in a tree-like fashion to respect the backtracking nature of web exploration. The backlink view is a simple list of page titles. While the page title is not always the most informative description of a page, it is the most easily accessible and is, most often, sufficiently useful. Navigation is simple: the titles in both panels behave like conventional hyperlinks, and load the page into the main browse window. Buttons to the left of the history tree let one view the backlinks of other pages in the tree.

At present only a maximum of 25 backlinks are displayed: this is partly a technical limitation, but one does find that most pages have fewer backlinks than this, so this number is sufficient. The backlinks are ordered by a simple heuristic: they are scored according to how many words in the backlink title appear in the target page, and sorted by this score. This works well at bringing good pages to the top of the list, without incurring the overhead of fetching the backlink pages themselves.

Of course, this prototype design has room for many enhancements. One might want to specify filters on what type of backlinks are displayed. One could add annotations and persistent storage to the history view, to build a bookmarking scheme. One could arrange for custom information to

[5] `http://www.netscape.com/download/index.html`

[6] `http://www.hotbot.com`

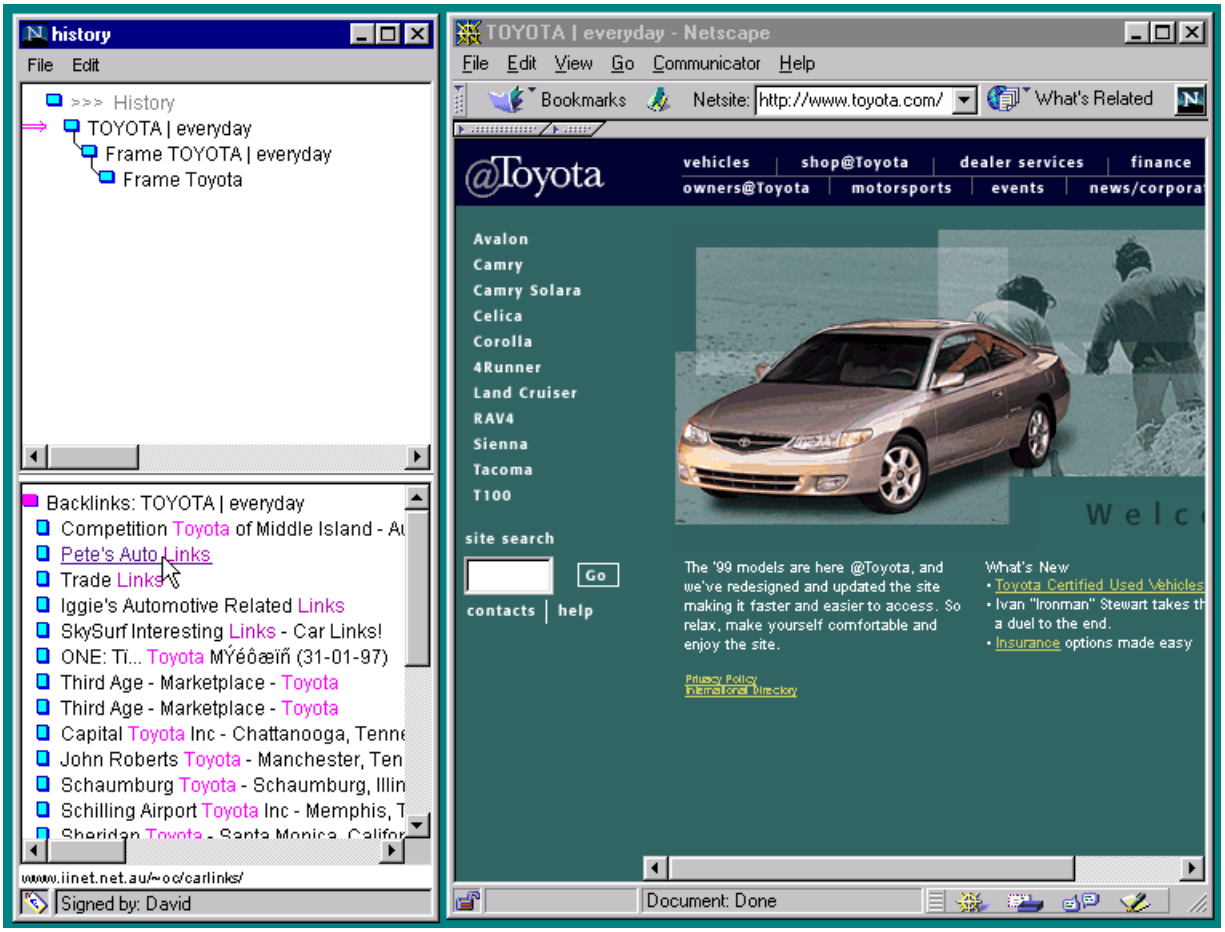

Figure 1: A screen-shot of the backlink browser applet. The browser assistant is showing the user's location in the upper panel (including frame substructure). The lower panel shows web pages that link to the Toyota home page. Some words that are related to the current page are highlighted. The user is about to click on the backlink shown by an arrow.

be displayed in place of simple backlink titles, or give the target page designer greater control over the form and content of the backlinks.

This design has been received favourably by early users. One interesting observation was that there seem to be two distinct ways of visualizing browsing activity: some users thought of it as a kind of backtracking tree-search, and some imagined it more as a "path in the woods". Both viewpoints are valid, and we now include a feature to flatten the history tree, for users who prefer to browse linearly.

2.2 Client implementation notes

We considered the following features such a browse assistant should ideally have. It should work with many browsers, since we don't want to have to recode it multiple times. It should be easy to install, without a hacked-up browser or patches. A plug-in is easier, but many users resist them. This led us to consider using Java. The assistant should also have a minimal performance impact on the client and the network.

The applet architecture is largely straightforward, but a few points should be made about the

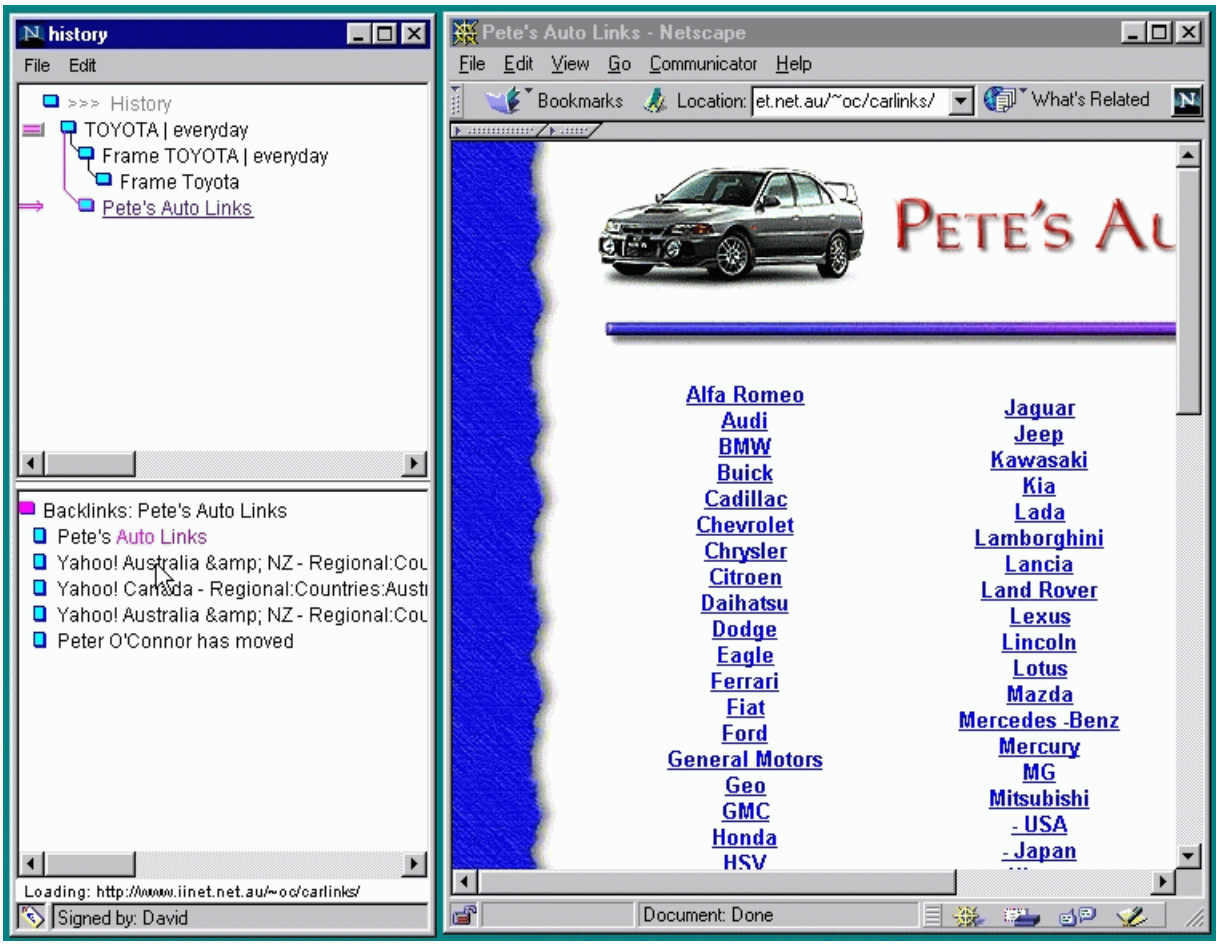

Figure 2: Screen-shot, continued. Clicking on the suggested backlink instantly takes the user to a extensive list of auto manufacturers around the world. (Some other backlinks are narrower, giving a range of generalizations of the topic of the initial page.) Also notice how the backlink traversal is recorded in the upper panel.

implementation details. The interface coding was constrained to be as lean as possible to speed loading times. We chose not to use Swing or other GUI toolkits, because the current generation of browsers does not include them by default. The applet performs several tasks which the Java default security model would disallow. It needs to be able to monitor the current browser window for new pages being loaded, fetch results from a search engine, and, for study purposes, we log the browse trail to local disk. In order to permit these operations, the applet must first be signed with a key certified by a trusted certification authority (CA) in order to certify authenticity. If the signature on the applet cannot be traced from a certificate in the browser, then it fails to run. For the purposes of our testing, we chose to produce our own self-signed certificate, and have the users download this as well. Once the applet starts, Netscape prompts the user for permission to perform the necessary operations. Similar mechanisms exist with Microsoft Internet Explorer, although the programming APIs are different and we restricted ourselves to a single implementation.

When the applet starts, it creates a new window, which should be used for further browsing. This means that the applet can continue to run uninterrupted in its own window, and that it can monitor the new window's browsing activity. Netscape's current model cannot send events to Java

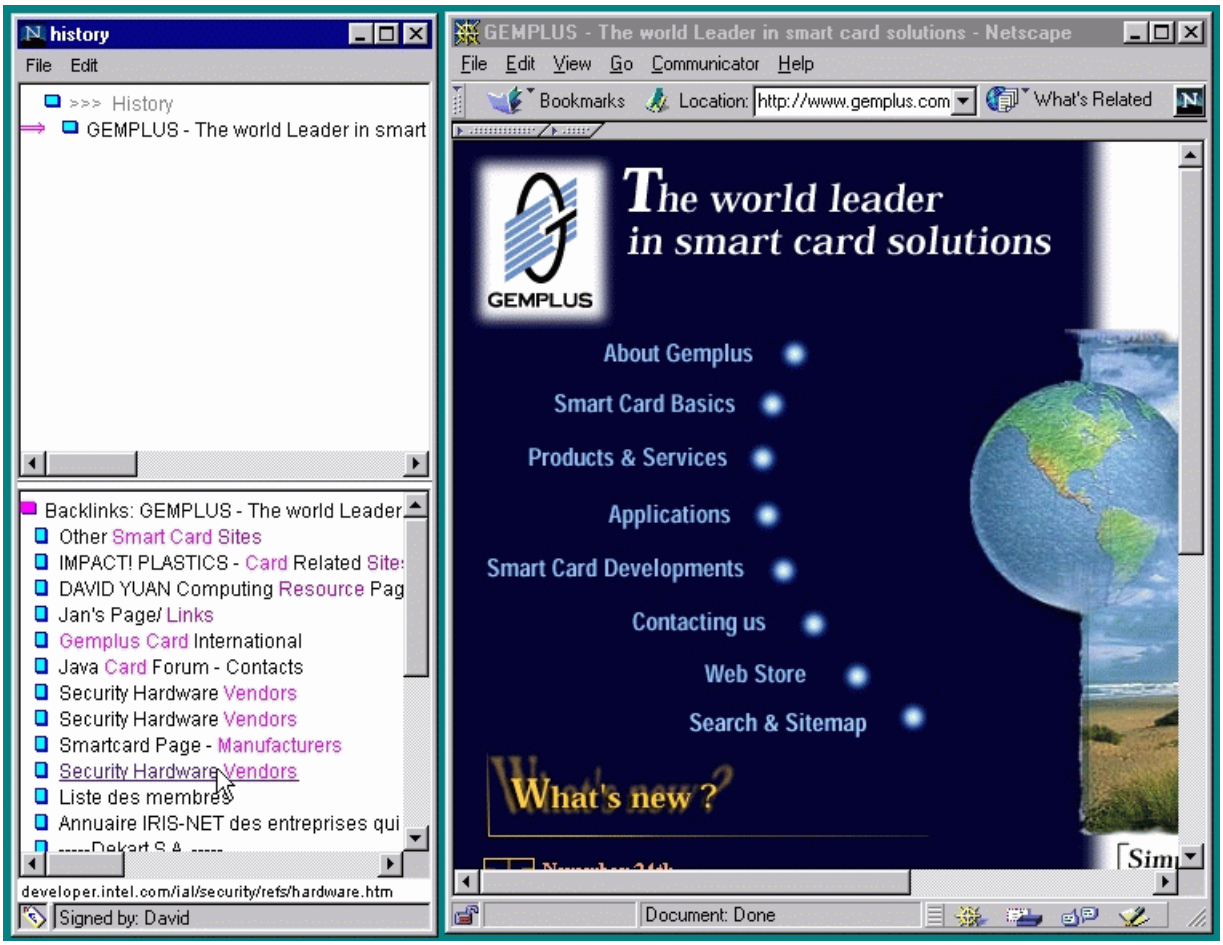

Figure 3: The browser is displaying the home page of Gemplus, the world's largest manufacturer of smart cards. Links from commercial sites to competitor sites are the exception rather than the rule, and here the browser assistant shows a backlink to an Intel site.

when a page is loaded, so the applet polls, by examining the contents of JavaScript variables. The applet can be configured to fetch backlinks from HotBot or AltaVista. An appropriate query string is formed, to minimize the amount of extraneous text returned, and the URLs and page titles are extracted from the engine results. Eventually it could also be configured to consult properly configured servers (see section 4).

2.3 User studies

We wanted to get some feel for the usefulness of a backlink browser assistant. Therefore we designed a user experiment. Given the diversity of the web as well as web users, the results should be regarded as anecdotal rather than rigorous. We picked some eight topics for exploration by our volunteers. Topics were chosen to have a sizable supporting community and yet narrow enough that some human browsing might be required over and above keyword searches and/or topic distillation [9]. The topics are shown in Table 1; they were made available online to volunteers [7].

Volunteers were assigned randomly to two of the topics in Table 1. Two versions of the browsing

[7] http://www.cs.berkeley.edu/~soumen/userstudy/UserStudy.html

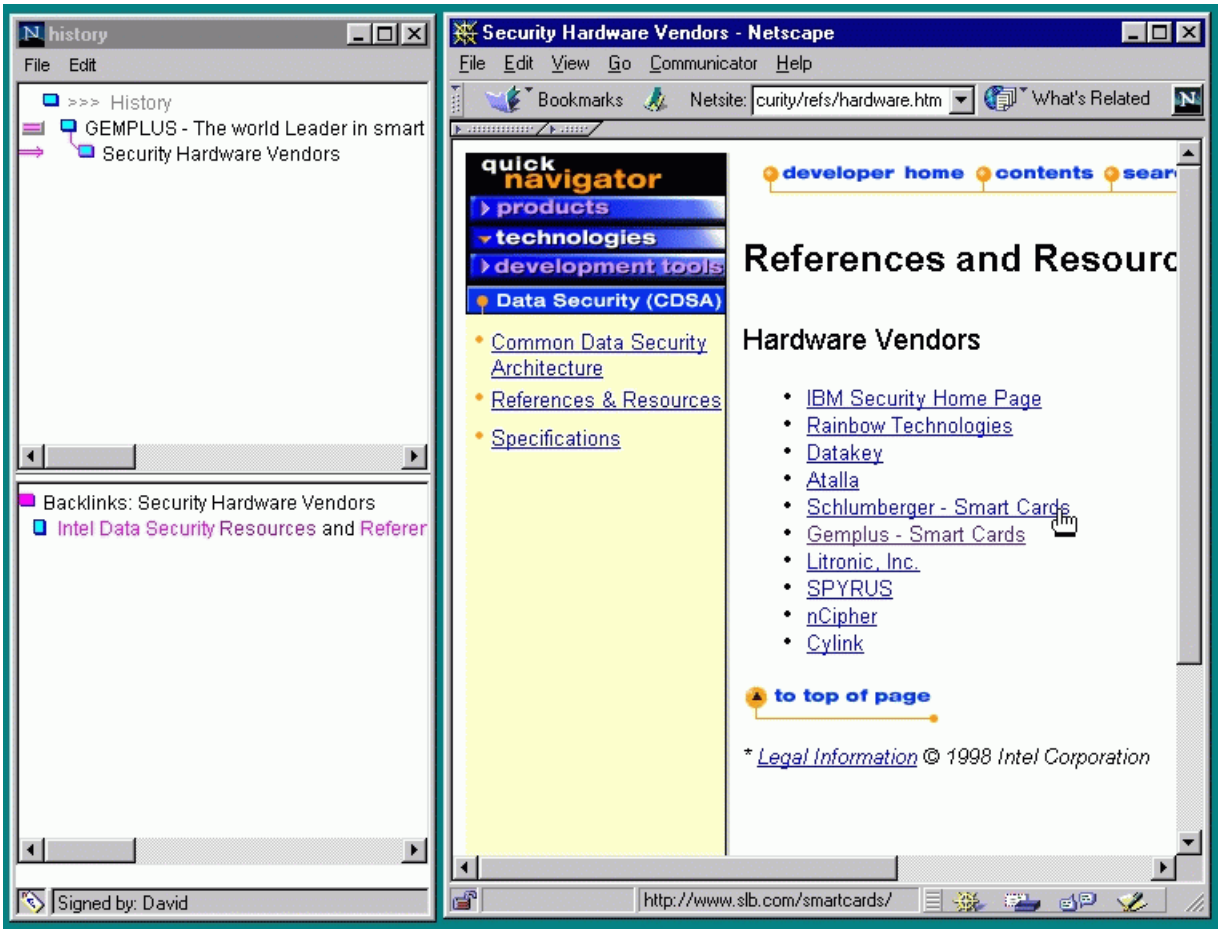

Figure 4: The user has navigated to the Intel page that points to Gemplus, and finds now a link to competitors of Gemplus. Following a path forward from Gemplus is less likely to lead to such a path.

assistant applet were provided to them. One version did not show backlinks ("placebo"), the other did. Each volunteer had to first search for one topic using the placebo applet. They would enter the search terms above to Alta Vista, then browse for 15 minutes, then stop and send us a log of all URL's visited as maintained by the applet. Then they would use the backlinked version of the applet, searching for the second topic for 15 minutes.

A Perl script stripped off the information as to which version of the applet each URL came from, and produced an evaluation form that was available online [8].

Finally, three people independently rated each URL without knowing the source, on a scale of 0 to 3. The scores are shown below.

[8] http://www.cs.berkeley.edu/~soumen/userstudy/

Code	Short name & Alta Vista query	Long description
A	Airbag safety `+airbag +injury +statistics`	Pages that compile statistics about airbag related injuries.
C	Curling equipment `+curling sport* +equipment`	Compilations of major manufacturers and distributors of equipment for the sport named curling.
G	vintage Gramophone equipment `+vintage +gramophone`	Information about different makes, models, and standards for old gramophone records and players.
L	Lyme disease `+"lyme disease" +statistics`	Statistics about Lyme disease occurrence over major affected regions.
M	leonid Meteor shower `+leonid +"meteor shower*"`	What will be possible effects of the meteor shower? What is the best way of watching it?
P	freedom of Press `+freedom +press +censorship`	Status and legislation related to censorhip and freedom of the press in various regions of the world.
S	Secure payment standards `+"secure payment" +standard*`	What are the major secure payment standards proposed or in use today?
T	Telecommuting `+telecommuting`	Organizations researching telecommuting and the benefits and disadvantages of telecommuting.

Table 1: Topics for user study.

Topic	Forward	Backlink	Both
A	22	148	11
C	68	42	5
G	10	112	2
L	34	22	14
M	54	131	11
P	65	24	3
Total	**253**	**479**	**46**

From these results it seems reasonable to conclude that the incorporation of backlink navigation into a browser produces some measurable improvement in the quality of information discovered on the web.

3 Extensions to HTTP

We began this project with the intent of building something quick and dirty to accelerate the use of backlink information for information retrieval. Our experiments with the browser assistant encouraged us to investigate server-side and protocol enhancements that would further the use of backlinks. Our design goals are:

- Clients should inform servers that they want backlink information.
- Servers should inform clients how and where they can retrieve backlink information.
- Proxies should to be able to assist clients in retrieving backlink information, possibly by aggregation from different sources.

- The protocol should be simple enough for embedded devices to implement a meaningful subset.
- It should be easy to integrate with the existing infrastructure.
- There should be zero or at least minimal performance impact for clients and servers that choose to ignore backlink information.

We considered several different methods for integrating the retrieval of backlink information into the browsing process. It could be delivered inline to the normal browsing process, or as a separate document retrieval. If delivered inline, it could be piggybacked on the normal HTTP [7, 12] process using keepalive to retrieve another document, or it could be delivered as part of a MIME multipart document, or it could be delivered in the HTTP response header. A client that wishes to take advantage of backlink information could express a willingness to receive the information in an `Accept:` header line, or negotiate the content that is delivered as in RFC 2295 [14]. None of these seem appropriate.

The original HTTP protocol [7] contained a method called "LINK" that was apparently intended for distributed maintenance of backlinks between resources. In addition, the `Link:` entity-header [12, Section 19.6.2.4] was suggested as a mechanism to convey backlink metadata in responses. The description sounds rather close to what we require, but unfortunately both were declared as experimental and unreliable in the HTTP 1.1 specification [12, Section 19.6]. As such they are now inappropriate foundations upon which to build.

Two more serious objections to the `Link:` entity-header are that it can degrade the performance of HTTP, and it fails to address several desirable features of backlink metadata. In particular, there is no transfer of ranking, freshness, or history data, and there is no way to issue queries on backlink metadata. The performance problem derives from the fact that backlink information is transferred in the header before transferring the actual resource body. Backlink metadata should properly be regarded as a secondary source of information about the resource, and fetching the metadata before fetching the resource is inefficient. Consider for example that HotBot already reports that it knows about 522,890 links to Yahoo!

WebDAV and RDF offer some attractive capabilities, but address far more than we needed and have a few drawbacks. In order to integrate the variety of sources that are available to clients and proxies, we chose to design an extension to HTTP that avoids some of these problems. Servers and clients that support WebDAV will be unaffected by the protocol we suggest.

3.1 HTTP Response Headers

In order for clients to be able to retrieve and display backlink information in a useful way, they need to establish the location and willingness to exchange such information. We propose several mechanisms to accomplish this. For clients and servers that support HTTP/1.0, the easiest way is for the server to indicate that backlinks are available, and supply a URI to tell the client where to fetch them. We chose to have the server report the availability of backlink information by including an optional end-to-end HTTP header response line formatted according to the HTTP Extension Framework [23]. It is perhaps easiest to specify an example, followed by the syntax description. A sample response header might look like:

```
Opt: "http://www.swcp.com/~mccurley/bl/"; ns=15-
15-location: text/html URI1
15-location: text/rdf URI2
```

Here each 15-location line indicates a MIME type and URI location for retrieving backlink information in that form. The URIs can be relative or absolute, to support both fetching the information from another site as well as from the server itself. The number 15 is merely an example, since it is dynamically generated by the server to avoid header conflicts as per [23]. The URIs should not be pointers to generic backlinks page, but rather should be specific to the page that was requested in the HTTP request.

If the client and server are using persistent connections, the client can go ahead and issue the request for backlink information into the pipeline as soon as they see the appropriate response-header indicating that backlinks are available. This streamlines the retrieval. If the client and server are not using persistent connections, or if the backlink information is to be retrieved from a different server, then a separate connection will have to be initiated by the client.

In our example, we gave two locations for backlink information, with two different MIME types for the content at those locations. The option of providing information in several forms is designed to support both collection by machine and interaction by humans. The `text/html` MIME type is intended for humans to view, and tailor their access to backlink information. This gives the server control over the display, search, access control, and policy. It may contain simply an HTML-formatted list of inlinks, or it may also contain a query interface for how to request links to the page. For example, a server may offer a query interface for requesting only inlinks from certain sites, or that are used with a certain frequency. It may also contain a human-readable statement of policy regarding availability of backlinks. In this case the design of the query interface is left to the server, but the server will still have information informing them of which URI backlink information is requested for.

One advantage to the HTML interface is that existing browsers can use it with minimal modification (a simple "backlinks" button may retrieve this and display it). For more sophisticated client programs, we believe that more elaborate and verbose methods based on RDF are appropriate. Servers that support WebDAV will require XML support, and can respond to a PROPFIND command with essentially the same response as an ordinary HTTP request for the referenced URI.

The description of response header lines follows [23], using lines of the form

```
prefix-location: URI mime-type [;charset]
```

where `prefix` is from [23], and `mime-type` is one of `text/html`, `text/rdf`, or a new MIME type called `x-backlinks/text` discussed below. The `charset` is optional, and follows the MIME specification. Additional MIME types are possible as well.

3.2 The `x-backlinks` MIME type

In the interest of simplicity for our Apache module, we devised a simple transfer mechanism for backlink information based on a new MIME type. The `x-backlinks` MIME type described in this section is intended to be easily parsed, human-readable, and compact. A response of this type consists of a sequence of lines ending in CRLF. Lines are of two types. First, lines of the form

```
Alt: mime-type URI
```

are used to specify further sources for backlink information. These can be used either to provide completely different sources, or in the case when more information is available from the server, a URI to fetch more or issue queries. URIs are site-specific, but examples include:

```
Alt: http://www.ibm.com/bl?uri=/contact.html x-backlinks/text
Alt: /bl?key=76x32 text/rdf; charset=us-ascii (Plain Text)
Alt: http://www.ibm.com/bl text/html
```

Second, there are lines specifying actual backlinks. These have the form:

`uri count first-date last-date title`

Each field except the URI can be replaced with a "`-`" character if the server chooses not to report it or data is unavailable. In addition, a line can be truncated at any point after the URI. Thus the simplest possible response consists of a URI. The meaning of each field is as follows:

`uri` the uri of the backlink. This may be either relative or absolute. Relative URIs are interpreted relative to the requested URI.

`count` the number of times that the referer has been received by the server during the period between `first-date` and `last-date`. If either of `first-date` or `last-date` is omitted, then the count is simply the total number of times that records are available for.

`first-date` the first date for which records are available. Both `first-date` and `last-date` are formatted according to RFC2068[12], and surrounded by the double-quote symbol ". Better efficiency would be achieved through the ISO 8601[1] format, but this has not gained wide acceptance in the world wide web.

`last-date` the date that the link was last referenced through a Referer field, or the last date that the link was validated by the server.

`title` a descriptive title for the uri. The origin and content of this text is unspecified, but is intended to describe to a human the relationship or content of the resource. It may be constructed by a variety of methods, including those described in section 1.4.1, other automatic processes, or manual human editing. In the event that a non-standard character set is used, it should be formated according to RFC 1522 [20].

The order of the individual backlink lines is left to the server to decide, and the client is free to use them as it wishes. The inclusion of frequency and freshness information is used to assist clients in customizing the display.

3.3 Proxies and Interoperability

In the event that both client and server support backlink information, the proxy should simply relay the requests, consistent with the existing caching and HTTP end-to-end extension mechanism. Proxies that are aware of other backlink information sources (e.g., search engines) may insert their own headers to inform the client that other sources are available.

Clients may be designed to only fetch backlink information upon the interest of the user, or they may retrieve and display backlinks as the default. We imagine that backlink information will be used only on infrequent user demand, and do not wish to impose an undue burden on the network.

The HTTP extensions that we propose here will be ignored by all compliant HTTP implementations. Implementations that do not support the extension will simply suffer a minor performance degradation from constructing and transmitting the headers. Proxies may recognize incompatibilities between servers and clients, and supply appropriate translations.

4 Server-side Implementation

Most of the server modification relates to the storage of the backlink database and query support for backlink-enabled clients. At the most rudimentary level, no special support is needed to provide

backlinks. The administrator simply turns on the referrer log option in any popular web server such as Apache[9], and installs a simple CGI script that `grep`'s the logs for the target URL. This will not satisfy our requirements of efficiency and frugality of space usage. At the other end, it is conceivable that fairly sophisticated database capability will be needed to support rather complicated queries being made to the server, e.g., "find all inlinks from a given site that have been followed at least 50 times in the last 10 hours."

We believe that web servers will continue to take on more and more characteristics of database servers in the future [11]. However, in the near term the greatest benefit for non-commercial sites of small to medium sites will probably come from the ability to make basic queries regarding inlinks. Moreover, avoiding dependence on a database will make it easier to distribute, upgrade and deploy.

We considered two candidates for integrating the backlink code with the server: modules and servlets. Apache, the most widely used server, provides a modular architecture by which additional functions can be integrated into the server through a standard API (Thau, [25]). The other option is to use server-side Java handlers, called *servlets*, to perform the necessary actions. With sufficient server-side capability to execute generic code associated with each page, a backlink server can be constructed with relatively little server customization. Some vendors also provide server-side scripting languages which (while mostly used for site maintenance) can be used to implement a backlink server [22].

We implemented the simple HTTP protocol extension as a module in the Apache web server. Our module performs three places in the Apache flow of processing. First, the module reports referer fields to a database, and keeps track of the number of times a link has been followed as well as the last time it was followed. Second, the module registers a special URL with the server, and supplies inlink information for urls on the server. Third, the server modifies outgoing response headers to show availability of backlink information. There are a number of configuration options, including:

filtering certain referers are automatically ignored by the server (e.g., `file:` or `javascript:` URLs). In addition, referers from other classes of URLs can be configured to be ignored or required by the server using the `RefererFilter` configuration command. The syntax is

```
RefererFilter ignore|require host|path expression
```

For example, consider the following lines:

```
RefererFilter ignore host *.almaden.ibm.com
RefererFilter require host *ibm.com
```

The first will direct the server to ignore referers with domains ending in almaden.ibm.com (matching is case-insensitive for hosts). The second line will direct the server to only log referers that come from the www.ibm.com host. Note that specifying a domain rather than IP address will require the server to have resolved the address, which could have a significant performance impact.

The following will direct the server to only log referers whose pathname contains the string "library":

RefererFilter require path *library*

Filters are applied sequentially, and the first matching filter is the one that determines the action.

[9] http://www.apache.org

backlink prefix the default configuration of the server will provide inlink information in response to a url of the form `http://server/_bl?url=target`. The backlink prefix string _bl can be configured as something else using the `RefererLocation` configuration command.

logging location The location of the database files can be specified using the `RefererLogPath` configuration command.

space saving The maximum size of the database can be configured using the `RefererDBSize` configuration command.

Rather than use a full-fledged relational engine behind the server, we decided to use a pared-down storage manager: the Berkeley DB[10] library, in order to encourage widespread dissemination with Apache. The current version of the database is implemented with three Berkeley DB hash tables. Keys into the tables are constructed by hashing URLs to 64-bit values. The three tables are constructed as follows:

1. a hash table where the keys are hashes of urls, and the values are the URLs themselves. This allows reconstructing the URLs from the keys.

2. a hash table where keys are hashes of URL pairs (referer,target) and data elements are (timestamp,count) pairs. This facilitates fast updates to the database and limited selective retrieval.

3. a multi-valued hash table where the keys are hashes of URLs, and the data elements are lists of inlinks, given by 64-bit hashes.

If the total number of URLs that a server encounters is below 2^{32}, and if the hash function behaves as a random function, then collisions between URLs are unlikely for the hash function. In the event that collisions occur, the only consequence is that a few stray backlinks may be supplied to users. Given the dynamic nature of the web, these stray backlinks will likely have little effect compared to the dead links that users regularly encounter.

Further development

One problem that we have not addressed completely is that of "spamming". If a server is open to suggestion about what sites should be listed as backlinks, then we can expect sites to use backlinks as a means to lure browsers to their pages. Consider for example an advertising page that contains "hidden" links to a site in the form of white printing on a white background, containing HREF tags. A server that becomes aware of such a link may automatically provide this as a backlink to clients, even though the advertising page has little to do with the server. If there are a large number of backlinks from the server page, then it becomes a competition for the advertising page to have their backlink listed ahead of others. Any automated procedure for ranking the pages is potentially vulnerable to abuse by the advertiser, since they may skew their content to raise their ratings. If the server uses frequency and number of distinct client IP addresses reporting a referer as an automated ranking function, then advertisers are faced with a difficult task since they must also induce users to follow the link out of their advertising page to the target page in order to reinforce their backlink rating on the target page. Alternatively, the advertisers would have to simulate clients following the link, which is consumes many resources. Thus it seems that there are ranking policies that will easily deter such an attack. Our server implementation implements

[10] http://www.sleepycat.com

a strategy of supplying backlinks ranked by frequency*diversity, making it unlikely that the server will be subject to the attack.

One can also enforce tighter checks, by imposing more flexible filters on the backlink sets, or even by such devices as displaying only backlinks which link to more than one specified target page. We leave this to future versions.

Our server implementation is designed to simply compile backlink data from the most readily available source, but clearly there is more that could be done. Our database is populated with an access count, timestamp of last access, and URL. In addition, the server could store a full history of access, textual information, ratings, rate of change, ownership, copyright information, or many other factors. This information might be managed through a variety of tools, including crawling, automated reporting and postprocessing, or human editing. These may prove particularly useful for sites that wish to implement a specific policy involving backlink metadata. For example, the server may wish to only report backlinks that are still active, or report all links that ever existed.

At present the Apache module only supports retrieval of all links to a given page. Queries about backlink information will require more of the functionality of traditional database servers. DASL looks like a promising means to support queries on server resources, including metadata. The exact use of such queries in a client is unclear however, and the most likely mechanism may remain a traditional HTML form interface for some time to come.

5 Conclusions

It is our firm belief that backlink metadata provides a significant enhancement of information gathering from web resources. We have described a protocol by which such data can be retrieved and used by clients, and some prototype tools that will support the infrastructure. These tools support only a minimal set of policies for *managing* the backlink metadata, and it is natural to expect that others will develop tools to further assist in the management and use of such data. For example, the integration of backlink data into browsing tools can be done in several ways, and we encourage software developers to think of the best ways to organize and present such information.

Acknowledgements: Thanks to our volunteers, Dimitrios Gunopulos, Inderjit Dhillon, Dharmendra Modha, Myron Flickner, Jason Zien, Kiran Mehta, Louis Nagtegaal, Daniel Oblinger, Amit Somani, SriGanesh Madhavan, Sunita Sarawagi, and Martin van den Berg. Special thanks to Sunita Sarawagi and Martin van den Berg for help with the URL evaluation. Thanks to Rob Barrett and Robert Schloss for helpful discussions.

References

[1] Data elements and interchange formats – information interchange – representation of dates and times. ISO standard 8601:1988.

[2] HTML 4.0 specification. Online at `http://www.w3.org/TR/REC-html40/`.

[3] Resource definition framework. Online at `http://www.w3.org/RDF/`.

[4] DAV searching & locating, July 1998. IETF Working Group note, online at `http://www.ics.uci.edu/pub/ietf/dasl/`.

[5] V. Balasubramanian. State of the art review on hypermedia issues and applications. Online at `http://eies.njit.edu/~333/review/hyper.html`.

[6] D. Bearman. Dublin core relation element working draft 1997-12-19. Online at http://purl.oclc.org/dc/documents/working_drafts/wd-relation-current.htm.

[7] T. Berners-Lee, R. Fielding, and H. Frystyk. RFC 1945: Hypertext Transfer Protocol — HTTP/1.0, May 1996. Online at http://www.cis.ohio-state.edu/htbin/rfc/rfc1945.txt.

[8] K. Bharat, A. Broder, M. Henzinger, and P. Kumar. The connectivity server: fast access to linkage information on the web. In *Seventh International World Wide Web Conference*, Melbourne, 1997. Online at http://www.research.digital.com/SRC/personal/Andrei_Broder/cserv/386.html.

[9] S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagopalan, D. Gibson, and J. Kleinberg. Automatic resource compilation by analyzing hyperlink structure and associated text. In *Proceedings of the 7th International World Wide Web Conference*, volume 30 of *Computer Networks and ISDN Systems*, pages 65–74, Brisbane, April 1997. Elsevier. Online at http://www7.scu.edu.au/programme/fullpapers/1898/com1898.html.

[10] K. E. Drexler. Hypertext publishing and the evolution of knowledge. Online at http://www.foresight.org/WebEnhance/HPEK0.html.

[11] M. Fernandez, D. Florescu, J. Kang, A. Levy, and D. Suciu. Catching the boat with strudel: Experiences with a web-site management system. In *SIGMOD Conference*. ACM, 1998.

[12] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee. RFC 2068: Hypertext Transfer Protocol — HTTP/1.1, Jan. 1997. Online at http://www.cis.ohio-state.edu/htbin/rfc/rfc2068.txt.

[13] Y. Goland, J. E.J. Whitehead, A. Faizi, and S. C. D. Jensen. HTTP extensions for distributed authoring – WEBDAV, November 1998. This online document at http://www.ietf.org/internet-drafts/draft-ietf-webdav-protocol-10.txt is an Internet-Draft. It is for informational purposes only and should be considered as work in progress.

[14] K. Holtman and A. Mutz. RFC 2295: Transparent content negotiation in HTTP, Mar. 1998. Online at http://www.cis.ohio-state.edu/htbin/rfc/rfc2295.txt.

[15] T. Kaehler. The backlinks page: Web enhancement project, June 1997. Online at http://www.foresight.org/WebEnhance/backlinks.news.html.

[16] Y. Maarek, M. Jacovi, M. Shtalhaim, S. Ur, D. Zernik, and I. B. Shaul. WebCutter: A system for dynamic and tailorable site mapping. In *World Wide Web Conference*, volume 6, Santa Clara, Apr. 1997.

[17] P. P. Maglio and T. Matlock. Metaphors we surf the web by. In *Workshop on Personalized and Social Navigation in Information Space*, Stockholm, Sweden, 1998.

[18] M. Marchiori. The limits of web metadata, and beyond. In *Proceedings of the 7th International World Wide Web Conference*, pages 1–9, 1998. Online at http://www.elsevier.nl/cas/tree/store/comnet/free/www7/1896/com1896.htm.

[19] R. Miller and K. Bharat. SPHINX: A framework for creating personal, site-specific web crawlers. In *World Wide Web Conference*, volume 7, Brisbane, Australia, Apr. 1998.

[20] K. Moore. MIME (multipurpose internet mail extensions) part two: Message header extensions for non-ascii text, September 1993. Online at http://www.cis.ohio-state.edu/htbin/rfc/rfc1522.html.

[21] T. Nelson and J. Walker. Project xanadu. Online at http://www.xanadu.net/ and http://www.xanadu.com.au/.

[22] NeoSoft. Neowebscript, Nov. 1998. Online at http://ftp.neosoft.com/neowebscript/index.html.

[23] H. F. Nielsen, P. Leach, and S. Lawrence. HTTP extension framework for mandatory and optional extensions, Aug. 1998. Internet-Draft, online at http://www.w3.org/Protocols/HTTP/ietf-http-ext/draft-frystyk-http-extensions-00.

[24] J. Slein, F. Vitali, E. Whitehead, and D. Durand. Requirements for a distributed authoring and versioning protocol for the world wide web, February 1998. Online at http://www.cis.ohio-state.edu/htbin/rfc/rfc2291.txt.

[25] R. Thau. Design considerations for the Apache server API. In *Fifth International World Wide Web Conference*, Paris, 1996. Online at http://www5conf.inria.fr/fich\protect\unhbox\voidb@x\kern.06em\vbox{\hrulewidth.3em}html/papers/P20/Overview.html.

[26] J. Walker. Hack links, July 1995. Online at http://www.fourmilab.ch/documents/hacklinks.html.