Website Reconstruction using the Web Infrastructure

[Extended Abstract]

Frank McCown
Old Dominion University
Computer Science Department
Norfolk, Virginia, USA 23529
fmccown@cs.odu.edu

ABSTRACT

Backup or preservation of websites is often not considered until after a catastrophic event has occurred. In the face of complete website loss, webmasters or concerned third parties may be able to recover some of their website from the Internet Archive. Other pages may also be salvaged from commercial search engine (SE) caches if caught in time.

We introduce the concept of "lazy preservation"- digital preservation performed as a result of the normal operations of the Web infrastructure (search engines and caches). We will investigate methods of how websites can be automatically reconstructed from the Web infrastructure (WI) by using a *web-repository crawler*. We propose to evaluate and measure the effectiveness of various web-repository crawler strategies and evaluate various methods for injecting the generative functionality (e.g., CGI programs, databases, etc.) of websites into the WI. We also propose to develop methods for tracking resources as they move through the WI and to characterize SE caches using random sampling.

1. INTRODUCTION

Digital preservation has been widely acknowledged to be an important problem with few easy solutions. Most preservation projects to date typically rely on *refreshing* (copying data onto newer media or systems), *migration* (transferring data to newer system environments) [41] and *emulation* (replicating the functionality of an obsolete system) [34]. Each of these strategies may involve a large institutional investment of time and money focused on preserving collections of known importance. For example, the Library of Congress recently funded a $13.9M project through the National Digital Information Infrastructure and Preservation Program (NDIIPP) in order to preserve "digital content relating to important people, events and movements that have had a major impact on the nation's history" [18]. We call these types of efforts *in vitro* digital preservation because of the limited scope, significant effort and controlled environments necessary for their success.

JCDL '06 Chapel Hill, NC USA

Other collections of unknown value may also be preserved although in a somewhat random and half-hazard manor by the "living web". For example, consider a 1994 NASA report (Figure 1) that has been refreshed and migrated to other formats (PDF, PNG, HTML) other than its original compressed PostScript. Only one of the links points to the original 1994 location. This type of *in vivo* preservation [12] which relies on the living web is not guaranteed by any single institution or archive; it is rather the result of distributed efforts of missions of users, web administrators and commercial services.

A major contributor to in vivo preservation is the Web infrastructure (WI), the collection of commercial web search engines (e.g., Google, Yahoo, MSN, etc.), personal web archives (e.g., Furl.net and Hanzo:web), web archives operated by non-profit companies (e.g., the Internet Archive's "Wayback Machine") and research projects (e.g., CiteSeer and NSDL). The WI supports refreshing and migrating of web resources, often as side-effect of their intended purposes. Some members of the WI that are in competition with each other (such as Google, MSN and Yahoo) to increase their holdings will continue to improve the utility of the WI. Although members of the WI may come and go, the combined efforts of the WI can ensure long term preservation of many web resources.

We introduce the concept of *lazy preservation*- in vivo preservation for websites that results from the normal operations of the WI. We are aware of several instances when lazy preservation was used for website reconstruction when backups did not exist:

- The Internet Archive's Wayback Forum contains several postings from individuals who have lost their website and had no backups [33]. They were wanting to restore their websites from the Internet Archive.
- The WWW 2006 conference website was temporarily inaccessible after a fire destroyed the Mountbatten building at the University of Southampton (UK) [10]. The conference organizers ran a script to recover some pages from the Google cache.
- In Dec 2005, Aaron Swartz created a tool called arcget to recover a website from the Internet Archive that had recently gone out of service [39]. Swartz was unaware of our work at the time.

We suspect that lazy preservation will continue to be needed. Although many organizations have the resources

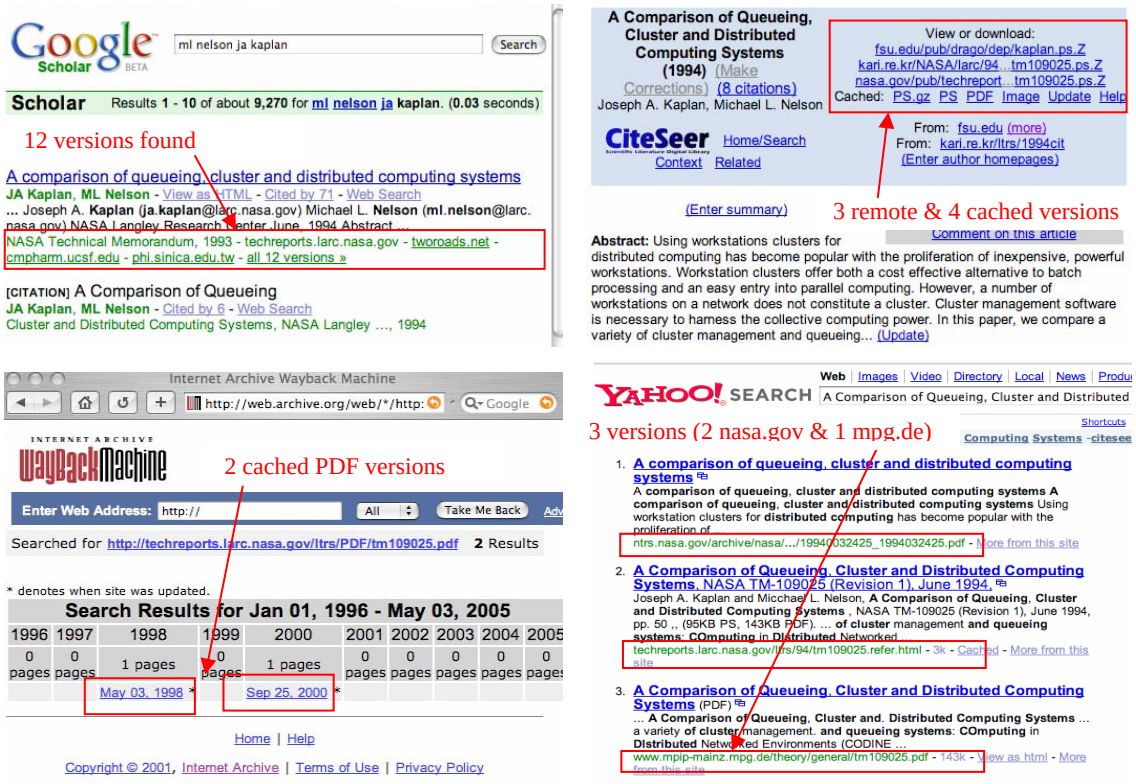

Figure 1: Refreshing and migrating occurring in the living Web

to perform backups, many individuals do not have the resources or only think of backups *after* they have lost all their data. It is not uncommon for web hosting services to tell their customers: "ultimately you are responsible for backing up your own data [6]." When a website ceases operation due to death of the owner, lack of funding or other calamity, third parties may rely on lazy preservation as the sole source for website reconstruction.

We propose investigating the use of the WI for reconstructing websites that have gone missing. Specifically, we plan to investigate the following questions:

- How completely can websites be reconstructed from the WI? (Section 4.2)
- What factors contribute to the success of website reconstruction? (Section 4.2)
- Which members of the WI are the most helpful for website reconstruction? (Section 4.2)
- What methods can be used to recover the server-side components of websites from the WI, and can such methods be successfully and effectively implemented? (Section 4.3)
- How can we track web resources as they are refreshed and migrated through the WI? (Section 4.4)
- What resources are typically stored in SE caches, and what factors contribute to resources not being cached? (Section 4.5)

2. BACKGROUND AND RELATED WORK

The ephemeral nature of the Web has been widely acknowledged. Even the casual Web user is familiar with web pages going 404 (not being found). Koehler [16] provides possibly the longest continuous study of URL persistence using the same set of 361 URLs randomly obtained in December 1996. Other researchers have measured linkrot (inaccessible URLs) based on subject matter [40] and their use in academic citations [23] and digital libraries [26].

Many solutions to missing web resources have been offered. Berners-Lee [4] has developed popular guidelines for creating durable URLs, and indirection mechanisms like Persistent URLs (PURLs) [36], handles [20] and Digital Object Identifiers (DOIs) [27] have been developed to increase URL longevity. Phelps and Wilensky [28] proposed the use of *robust hyperlinks* to find missing resources by using lexical signatures, and Harrison [12] developed a system called Opal which uses lexical signatures to aid users in finding missing web resources from the WI.

These solutions focus on resources that change locations on the Web. To combat web resources disappearing altogether, researchers have typically focused on building web archives. Brewster Kahle founded the Internet Archive (IA) in 1996 as the first large-scale attempt to create an archive of the publicly accessible Web [14]. Other institutions and governments like the UK [2], Sweden [17] and Greece [19] have since recognized the value of preserving portions of the Web and are actively engaged in archiving culturally important websites.

Commercial services like FURL (`furl.net`) and Spurl.net

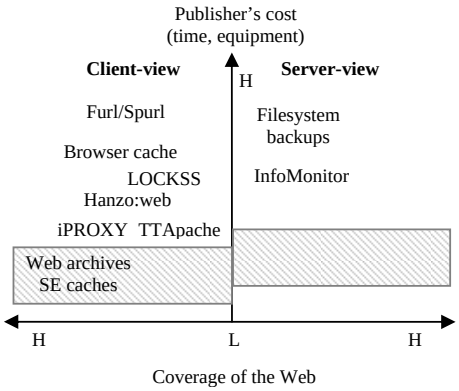

Figure 2: Publisher's cost and Web coverage of preservation systems/mechanisms

(`spurl.net`) have emerged recently that allow users to archive selected web resources. Hanzo:web (`hanzoweb.com`) is a similar service that easily allows an entire website to be archived.

Other non-commercial systems have addressed archiving individual websites and web pages. LOCKSS [32] was created to archive selected publishers' websites for library use. InfoMonitor archives the server-side components (e.g., CGI scripts and datafiles) and filesystem of a web server [7]. Other systems like the Apache module TTApache [8] and the proxy server iPROXY [31] archive requested pages from a web server but not the server-side components.

In regards to commercial SEs, the literature has mostly focused on measuring the amount of content they have indexed (e.g., [24]) and determining how well they respond to users' queries (e.g., [22]). Lewandowski et al. [21] studied how frequently Google, MSN and Yahoo updated their cached versions of web pages, but we are unaware of any research that attempts to measure how quickly new resources are added to and removed from commercial SE caches. As far as we aware, our work [25] was the first to focus on the use of SE caches for digital preservation.

Each of the Web preservation systems previously mentioned can be divided into two categories based on the level of preservation they provide:

1. *client view* - This is the view the client is presented after making an HTTP request for a web resource.

2. *server view* - These are the generative mechanisms or server components (scripts, files, databases, etc.) that are responsible for generating the client view of the website.

Preserving the client view is generally sufficient for those who are interested in what a particular website said or looked like. Recovering the server view is often more important for a webmaster that needs to recover the functionality of a lost website. Of all the systems previously mentioned, only InfoMonitor and filesystem backup mechanisms are designed for recovering the server view.

The graph in Figure 2 maps the systems mentioned above according to the website publisher's relative cost (in time, effort or equipment) to have his or her website preserved and the coverage that the systems provide to the entire Web. Furl and Spurl can be used to preserve much of the Web, but they are not practical for an individual archiving an entire website. A browser cache will store some requested resources, but it is impractical for a user to browse every resource in a website. Other systems are dedicated to preserving content from a particular website (TTApache, InfoMonitor, filesystem backup) or a small group of sites (LOCKSS). Systems like iPROXY may be configured to crawl and archive multiple websites, but they cannot cover nearly the breadth that the WI covers.

The shaded box of Figure 2 shows that search engines and web archives provide wide coverage of digital preservation services for the client view, but there is no such service provided for preservation of the server view. Lazy preservation (LazyP) leverages SE caches and web archives to provide high preservation coverage of the Web, and it requires no work for the website creator except that the website contents be accessible to a crawler (usually the de facto behavior).

We propose to find an equitable solution for reconstructing the server-side components of a website. For just a little more effort, lackadaisical preservation (LackP) can be used to support preservation of the server view. LackP will likely require the publisher to install an Apache module and set some configuration information in order to tell the module which files may and may not be injected into the WI. Once this installation and configuration is completed, there is no need for the publisher to purchase additional hard drive space, monitor backups and perform administrative duties to keep their website backed-up.

3. PRELIMINARY WORK

We have performed a preliminary investigation into using the WI for website reconstruction in [25] and give further details in [37]. Our work focused on examining the caching behavior of search engines on several decaying web collections and on demonstrating the feasibility of website reconstruction from the WI using a web-repository crawler.

3.1 Search Engine Caching

Search engines like Google, MSN and Yahoo provide access to their cached resources in case the original resource is temporarily inaccessible. Text-based resources like PDF, Word documents, Excel spreadsheets, etc. are often converted into HTML before being made available in the SE cache. Images are cached as thumbnails. Unlike SEs, IA stores all resources in their canonical form.

In order to measure how quickly resources enter and leave the caches of Google, MSN and Yahoo, we created 4 synthetic web collections consisting of a few hundred HTML, PDF and image resources. The web collections were deployed in June 2005 at four different locations. We provided links to the web collections from previously crawled pages in order to entice the indexing of our web collections.

We deleted resources from the collections on a daily basis for 90 days until most of the collections were gone. We examined the server logs to determine when the resources were crawled and performed daily queries to each SE to determine when the resources entered and left the SE caches.

From a website reconstruction perspective, Google outperformed MSN and Yahoo in nearly every category. Google cached the highest percentage of HTML resources (76%) and took only 12 days on average to cache new resources from three of the web collections. On average, Google cached

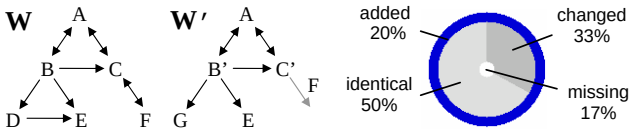

Figure 3: Lost website (left), reconstructed website (center) and reconstruction diagram (right)

HTML resources for the longest period of time (76 days), consistently provided access to the cached resources (86%), and were the slowest to remove cached resources that were deleted from the web server (51 days). Although Yahoo cached more HTML resources and kept the resources cached for a longer period than MSN, the probability of accessing a resource on any given day was only 53% compared to 89% for MSN.

3.2 Reconstructing Websites

A website can be represented as a graph $G = (V, E)$ where each resource r_i (HTML, PDF, image, etc.), identified by a URI, is a node v_i, and there exists a directed edge from v_i to v_j when there is a hyperlink or reference from r_i to r_j. The left side of Figure 3 shows a web graph representing some website W if we began to crawl it beginning at A. Suppose W was lost and reconstructed forming the website W' represented in the center of Figure 3.

For each resource r_i in W we may examine its corresponding resource r'_i in W' that shares the same URI and categorize r'_i as *identical* (r'_i is byte-for-byte identical to r_i), *changed* (r'_i is not identical to r_i), or *missing* (r'_i could not be found in any web). We would categorize those resources in W' that did not share a URI with any resource in W as *added* (r'_i was not a part of the current website but was recovered due to a reference from r'_j).

Figure 3 shows that resources A, G and E were reconstructed and are identical to their original versions. An older version of B was found (B') that pointed to G, a resource that does not currently exist in W. Since B' does not reference D, we did not know to recover it. It is possible that G is actually D renamed, but we do not test for this. An older version of C was found, and although it still references F, F could not be found in any web repository.

A measure of change between the lost website W and the reconstructed website W' can be described using the following **difference vector**:

$$\text{difference}(W, W') = \left(\frac{R_{changed}}{|W|}, \frac{R_{missing}}{|W|}, \frac{R_{added}}{|W'|} \right) \quad (1)$$

For Figure 3, the difference vector is (2/6, 1/6, 1/5) = (0.333, 0.167, 0.2). The best case scenario would be (0,0,0), the complete reconstruction of a website. A completely unrecoverable website would have a difference vector of (0,1,0).

The difference vector for a reconstructed website can be illustrated as a **reconstruction diagram** as shown on the right side of Figure 3. The changed, identical and missing resources form the core of the reconstructed website. The dark gray portion of the core grows as the percentage of changed resource increases. The hole in the center of the core grows as the percentage of missing resources increases. The added resources appear as crust around the core. This representation was used in our website reconstruction experiments reported in [25].

3.3 Web Repository Crawler

We developed a web-repository crawler called Warrick to automate the reconstruction of websites from IA, Google, MSN and Yahoo. In August 2005 we downloaded 24 handpicked websites that varied on the basis of top-level domain name, size, subject and file type makeup. All files needed to produce each web page were downloaded (HTML, style sheets, external JavaScript files, images, etc.). Immediately after downloading the websites, we ran Warrick to reconstruct all 24 websites and compared the downloaded sites with the reconstructions.

We were able to recover more than 90% of the original resources from a quarter of the 24 websites. On average we were able to recover 68% of the website resources (median=72%). Of those resources recovered, 30% of them on average were not byte-for-byte duplicates. A majority (72%) of the 'changed' text-based resources were almost identical to the originals (having 75% of their shingles [5] in common). 67% of the 24 websites had obtained additional resources when reconstructed which accounted for 7% of the total number of resources reconstructed per website.

A majority (92%) of the resources making up the original websites were HTML and images. We were much more successful at recovering HTML resources than images; we recovered 100% of the HTML resources for 9 of the websites (38%).

Google contributed the most to each website reconstruction, providing on average 44% of the resources to each website and failing to contribute to only one website. MSN was second, providing on average 30% of the resources; IA was third with 19%, and Yahoo was last with a 7% contribution rate.

4. PROPOSED WORK

4.1 Warrick Enhancements

After our preliminary work using Warrick to reconstruct 24 websites, we recognized several improvements that could be made to Warrick to reduce the number of issued queries and improve URL normalization.

Search engines often implement a search parameter called 'site:' which tells a SE to return all the URLs it has indexed. IA may also be queried using an undocumented feature to list all stored URLs for a particular website. We shall call these types of queries *lister queries*.

There are two advantages of using lister queries: 1) discovery of URLs that we may not be aware of by just examining links in recovered pages, and 2) avoiding unnecessary queries when we know a particular resource is not stored in a web repository.

The use of lister queries allows us to reconstruct a website using one of three web crawling policies:

1. **Naïve Policy** - Reconstruct without performing lister queries.
2. **Exhaustive Policy** - Recover all resources that are produced by lister queries.
3. **Knowledgeable Policy** - Recover only resources discovered through page scraping, but use lister queries to prevent unnecessary web repository queries.

We would like to test these crawling policies on the 24 websites that we used in our preliminary reconstructions to see what effects it might have on our reconstructions.

There are other web repositories (e.g., Stanford's Webbase [13], Cornell's Web Laboratory [1], and hanzo:web) and SE caches (e.g., ask.com, gigablast.com, and incywincy.com) that could be useful for augmenting lazy preservation. We would like to reduce the work required to add additional web repositories to Warrick by developing a standard interface or API that could be implemented by a web repository. The repository could implement the interface itself or a binding could be implemented by interested third parties. Providing a standard API will lower participation constraints in lazy preservation.

4.2 Factors Influencing Reconstruction

Our preliminary experiment did not reveal any statistically significant results that would determine if a website's size or Google's PageRank would influence its recoverability from the WI. We would like to download and reconstruct a large number of randomly selected websites and perform regression analysis to determine if there are any factors that may determine the success of reconstructing a website.

4.2.1 Methodology

Instead of hand-picking the websites to be reconstructed, we need to select a random sample from the Web. One method would be to randomly sample from DMOZ (`http://dmoz.org`). We could download a suitable number websites and reconstruct them using the knowledge policy. For each website we would need to manually record the PageRank (using the Google toolbar). We would need to monitor the sites for several months to measure page change rates, and we would need to perform several reconstructions to study how they change over time.

4.2.2 Evaluation

After reconstructing the websites, a number of characteristics will need to be evaluated to determine if they contribute to the success of website reconstruction: PageRank, website size, MIME types, page depths, dynamic pages and page change rates. We would like to develop predictive models for determining how much of a website could be recovered from the WI if lost today.

We are also interested in determining which web repositories contributed the most resources to website reconstruction, and how many queries did web repositories need to handle versus their usefulness for website reconstruction.

4.3 Recovery of Web Server Components

4.3.1 Overview

Lazy preservation relies on web repositories which are focused on storing static website files or the static pages produced by server-side mechanisms. Dynamically generated pages produced by PHP, CGI, JSP, ASP and the like are increasingly common. Recovery of these server-side components along with the data files, databases and binary executables is necessary for complete website reconstruction. We call this lackadaisical preservation because it requires a little more effort than lazy preservation.

We would like to study several methods for exposing the server-side components to surface web crawlers so they may be crawled just as easily as static web pages are crawled. This could be as simple as exposing a compressed tar file to a web crawler, but SE crawlers do not crawl such files due to their lack of indexable content. We must instead investigate methods that could be used to inject server-side components into web pages that SE crawlers do crawl and cache.

Our preliminary research into examining website reconstruction revealed that we were most successful at recovering HTML resources. HTML resources have several characteristics that make them an excellent format for storing server-side components: 1) HTML resources are text-based unlike other popular web resources (e.g., images, PDFs, Word documents, etc.) that use binary formats, 2) all web repositories store canonical versions of HTML resources but usually not others, and 3) comments can be easily injected into HTML resources without affecting the overall appearance of the resource in a browser.

4.3.2 Injection Techniques

There are three techniques that could be used for injecting server-side components into HTML resources: 1) insert an entire encrypted server file into HTML comments, 2) divide an encrypted server file into smaller parts and insert each part into separate HTML files, and 3) compute erasure codes for an encrypted server file and inject the codes into HTML files.

Encryption is used to ensure that the server file is not readable to anyone but the publisher (security aspects are discussed in section 4.3.5). Method 1 may make HTML files too large for a web server to index. Search engines in the past have only indexed portions of large HTML files. Also, if the HTML file containing the server file was not recoverable, the server file would be lost.

Method 2 would allow us to create smaller HTML files which would have a higher likelihood of being cached. The problem with this method is that *all* HTML files containing the server file sections must be recovered to ensure the recovery of the server file. This would especially be important for binary server files where missing even one byte would make the file unusable. Method 3 fixes this problem by allowing complete recovery of the server file by only recovering a subset of the HTML files.

The technique for computing erasure codes, specifically Reed-Solomon coding, has been frequently used in RAID systems where portions of files are distributed among multiple devices to prevent loss of data [29]. They have also been used in secret-sharing systems [15] and for information dispersal [30].

4.3.3 Injecting Server Files into HTML Resources

Figure 4 shows an overview of how a server file could be injected into web pages and later be recovered from a SE cache. Steps a-d are performed when a server file is created or changed. Steps e-h are performed when needing to recover the server-side file.

a) Encrypt the file using a secret key known only to the webmaster or other trusted parties. The encrypted file will be base 64 encoded so it can easily be inserted into web pages as text.

b) Compute erasure codes for the encrypted file and break the file into n pieces (chunks) where recovery of any r

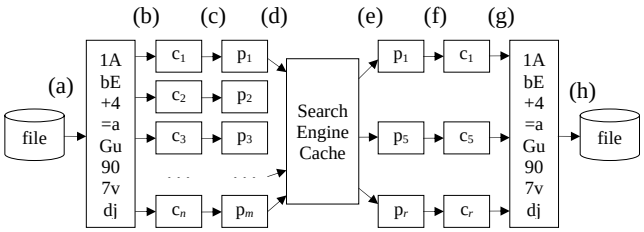

Figure 4: Recovery of encrypted server-side files from SE caches

chunks allows for complete recovery of the encrypted file.

c) Insert each chunk and metadata into m HTML files (where $n \leq m$).

d) Wait for SEs (or other web repositories) to crawl and cache the HTML files.

e) Recover as many HTML files from SE caches as possible.

f) Extract available chunks and metadata from the HTML files.

g) If at least r chunks have been recovered, reconstruct the encrypted file from the erasure codes.

h) Decrypt the encrypted file using the secret key.

Research needs to be performed to measure optimal values of n and r for a given website. The values of n and r are at odds with each other. Ideally we want to put as little encoded data into each HTML resource as possible (large values of n) but at the same time we want to minimize the number of chunks that we need to recover (small values of r) which requires n to be lower.

We need to research the values of n and r as a function of a website's indexable size. Websites with a small number of pages will require larger amounts of data per page. Larger websites will require less of a load per page.

4.3.4 Strategies for Resource Acquisition

We would like to investigate various methods for server-side injecting and measure their success of being stored by various web repositories. One method that could be used is to create "robot vaults", pages that are designed for only robots to crawl. These pages could be revealed to a crawler using cloaking or through a "robots only" link from a website's host page. The pages would consist of simple HTML along with the embedded portions of server files. We would need to carefully consider the design of the pages to make sure a SE would not reject the pages as spam. Changes to any server files would require automated updating of the robot vault to reflect the changes.

Another method would involve injecting the server file portions into "real" pages. Static pages that reside on the web server could be injected during low server activity periods. Or injections could be made dynamically when pages are being requested from the web server. The benefit of this method is that we would not run the risk of having our pages rejected as spam. We could run into problems though if the amount of available real pages to be crawled was very small and large server files had to be injected into a small number of pages. Such a consideration is not necessary in a robot vault where we can control the number of available pages to be crawled.

4.3.5 Security Considerations

There are several security-related concerns which must be addressed by our injection schemes. How will we keep server files secure from unauthorized access when they are injected into the publicly accessible WI? What server files are suitable for injected into the WI? How can server files be recovered if the password is lost or the webmaster who was responsible for the website has deceased or will not divulge the password?

4.3.6 Repository Antagonism

Cloaking, the serving of different content for web crawlers than for regular browsers, is a controversial topic [38]. Search engines often penalize web sites that employ cloaking (when detected) by removing websites from their index. If we employ injection of server encodings into web pages that are crawled by web repositories only, we risk the entire website being removed from the web repository. This issue will need to be studied further.

If many users adopt the server-side injection techniques we are proposing, search engines are likely to resent the extra load it places on their services. They may begin to use techniques to remove the encoding blocks from their repositories. We will therefore need to investigate methods that will allow us to "hide" encoded blocks in the HTML, to make it difficult for web repositories to automatically detect and remove the encoded blocks.

4.3.7 Evaluation

An effective evaluation of our system will involve the complete recovery of a website that uses a significant number of server-side components to function. We propose to create two websites using EPrints software [9] with a few hundred resources in each. One will use server-side injection techniques, and the other will not. We will attempt to reconstruct the websites on a weekly basis and compare the reconstructions. Once the "injection" site has been adequately crawled by at least one web repository, we would expect complete recovery of all server-side components.

4.4 Tracking Resources

4.4.1 Overview

For some resources that appear in the WI, it is trivial to determine where and when the resource was obtained. For example, a resource that appears in Google's cache can be mapped to an entry in a web server log where an http GET request was received from a Google-owned IP address with the word 'Googlebot' in the user agent field. Other web repositories may rely on third parties for web crawling. If the third party is not known, it can be difficult to correlate the request for a particular resource to the existence of the resource in a web repository. Sometimes the server logs are not even available to determine if a resource was crawled. We ran into this problem during our preliminary study when the server logs were lost for several days. Without such data,

it impossible to measure how quickly a resource has been crawled.

If a resource that is recovered from a web repository could reveal metadata about when it was crawled and by whom (IP address and user agent), there would be no need for analysis of web server logs to determine this information. We would like to explore the possibility of creating a mechanism for tagging resources as they are requested.

4.4.2 Methodology

We would like to develop a tagging mechanism (possibly as a server CGI or Apache module) that would intercept an http GET request and embed the request metadata directly into the resource before returning it in the http response. We could evaluate several strategies for injecting metadata into a resource: 1) inject the metadata directly into the resource in a human-readable form, 2) inject the metadata directly into the resource in a format that only we can understand, and 3) inject an identifier into the resource that can be mapped back to a log file entry containing the metadata.

4.4.3 Evaluation

In order to validate the effectiveness of our tracking mechanism, we could use it to verify the Search Engine Partnership Chart that is produced by Ihelpyou Inc. [35]. This complex chart shows which companies supply search results to a variety of search engines. Compiling such a chart is likely done by combing through publicly available information on company websites, through contact with individuals with inside information and by other evidence. The accuracy of such charts are difficult to verify.

We could produce a small number of tracking pages with an embedded globally unique identifier and perform daily queries to a variety of web repositories to see if they have indexed our pages. Each page could contain a "quote of the day" or other dynamically produced content so that it would be unique for each visitor. We could use the results of the repository queries to map the crawler-repository relationships that our data uncovers.

4.5 Characterization of Search Engine Caches

Although we know a lot about the archived material in the IA, we know very little about the type of content that is available in the SE caches; there has been no in depth analysis that we are aware of except the exploratory research we performed in [25]. Questions which need to be answered are:

- What percentage of resources that are indexed are also available in a SE cache?
- What are the main characteristics of resources (type, size, age) found in SE caches?
- Do the http Cache-control directives 'no-cache' and 'no-store' stop resources from being cached?
- How do different SE caches compare?
- How prevalent is the use of NOARCHIVE meta tags to keep HTML pages from being cached?

In order to answer these questions, we need to randomly sample URLs from a variety of SEs [3] and determine if the URLs are cached or not. We need to also request a live version of the resource to see how old the cached resource is (if that can be determined) and to see if there are any characteristics about the document that would keep it from being cached (e.g., use of the 'no-cache' directive, NOARCHIVE meta tag, etc.).

The results of our sampling could be combined with recent measurements of search engine size [11] to determine the total number of resources cached on the entire Web. We could also perform an overlap analysis with IA which has not been done before.

5. CONCLUSIONS

We have defined a new type of digital preservation, lazy preservation, which exists in the living Web. Lazy preservation is a low-cost, high-coverage form of preservation with no quality-of-service guarantees. The research outlined in this paper will allow us to qualitatively evaluate how much preservation is being performed for little or no work on behalf of the typical website. At the completion of this research, we will have demonstrated the lazy preservation technique, provided a reference implementation and characterized SE caching behavior. The major contribution of this work would be the successful implementation and evaluation of lackadaisical preservation which will allow the functionality of a web server to be recovered with little cost to the website maintainer.

We do not suggest that lazy and lackadaisical preservation are ideal preservation schemes, but they may be the only form of preservation available for third parties and in an emergency.

6. ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Michael L. Nelson, for his guidance in this research and for reviewing this abstract.

7. REFERENCES

[1] W. Y. Arms, S. Aya, P. Dmitriev, B. Kot, R. Mitchell, and L. Walle. A research library based on the historical collections of the internet archive. *D-Lib Magazine*, 12(2), Feb 2006.

[2] S. Bailey and D. Thompson. UKWAC: Building the UK's first public web archive. *D-Lib Magazine*, 12(1), 2006.

[3] Z. Bar-Yossef and M. Gurevich. Random sampling from a search engine's index. In *Proceedings from WWW '06*, pages 367–376, 2006.

[4] T. Berners-Lee. Cool URIs don't change. 1998. `http://www.w3.org/Provider/Style/URI.html`.

[5] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. *Computer Networks and ISDN Systems*, 29(8-13):1157–1166, 1997.

[6] Computer Tyme Web Hosting. `http://www.ctyme.com/hosting/`.

[7] B. F. Cooper and H. Garcia-Molina. Infomonitor: Unobtrusively archiving a World Wide Web server. *International Journal on Digital Libraries*, 5(2):106–119, April 2005.

[8] C. E. Dyreson, H. Lin, and Y. Wang. Managing versions of web documents in a transaction-time web server. In *Proceedings from WWW '04*, pages 422–432, 2004.

[9] Eprints. http://www.eprints.org/.

[10] Fire destroys top research centre. Oct 31 2005. http://news.bbc.co.uk/2/hi/uk_news/england/hampshire/4390048.stm.

[11] A. Gulli and A. Signorini. The indexable web is more than 11.5 billion pages. In *Proceedings from WWW '05*, pages 902–903, May 2005.

[12] T. L. Harrison. Opal: In vivo based preservation framework for locating lost web pages. Master's thesis, Old Dominion University, 2005. http://www.cs.odu.edu/~tharriso/thesis/writeup/THESIS_final.pdf.

[13] J. Hirai, S. Raghavan, H. Garcia-Molina, and A. Paepcke. WebBase: a repository of web pages. In *Proceedings of the 9th international WWW conference on Computer networks*, pages 277–293, 2000.

[14] B. Kahle. Preserving the Internet. *Scientific American*, 276(3):82–83, March 1997.

[15] E. D. Karnin, J. W. Greene, and M. E. Hellman. On secret sharing systems. *IEEE Transactions on Information Theory*, 29(1):35–41, 1983.

[16] W. Koehler. A longitudinal study of web pages continued: A consideration of document persistence. *Information Research*, 9(2), 2004. Available at http://informationr.net/ir/9-2/paper174.html.

[17] Kulturarw - Sweden's Web archive. 2006. http://www.kb.se/kw3/.

[18] G. Lamolinara. Library of Congress announces awards of $13.9 million to begin building a network of partners for digital preservation. 2004. http://www.digitalpreservation.gov/index.php?nav=4&subnav=2.

[19] C. Lampos, M. Eirinaki, D. Jevtuchova, and M. Vazirgiannis. Archiving the Greek Web. 4th International Web Archiving Workshop (IWAW'04), September 2004.

[20] L. Lannom. Handle system overview. In *66th IFLA Council and General Conference*, 2000. http://www.ifla.org/IV/ifla66/papers/032-82e.htm.

[21] D. Lewandowski, H. Wahlig, and G. Meyer-Beautor. The freshness of Web search engines' databases. *Journal of Information Science*, 32(2), 2006.

[22] F. McCown, J. Bollen, and M. L. Nelson. Evaluation of the NSDL and Google search engines for obtaining pedagogical resources. In *Proceedings from ECDL 2005*, pages 344–355, 2005.

[23] F. McCown, S. Chan, M. L. Nelson, and J. Bollen. The availability and persistance of web references in D-Lib Magazine. 5th International Web Archiving Workshop (IWAW'05), September 2005.

[24] F. McCown, X. Liu, M. L. Nelson, and M. Zubair. Search engine coverage of the OAI-PMH corpus. *IEEE Internet Computing*, 10(2), Mar/Apr 2006.

[25] F. McCown, J. A. Smith, M. L. Nelson, and J. Bollen. Reconstructing websites for the lazy webmaster. Technical report, Old Dominion University, 2005. http://arxiv.org/abs/cs.IR/0512069.

[26] M. L. Nelson and B. D. Allen. Object persistence and availability in digital libraries. *D-Lib Magazine*, 8(1), 2002.

[27] N. Paskin. E-citations: Actionable identifiers and scholarly referencing. *Learned Publishing*, 13(3):159–168, 2002.

[28] T. A. Phelps and R. Wilensky. Robust hyperlinks cost just five words each. Technical Report UCB/CSD-00-1091, EECS Department, University of California, Berkeley, 2000.

[29] J. S. Plank. A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems. *Software Practice and Experience*, 27(9):995–1012, 1997.

[30] M. O. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the ACM*, 36(2):335–348, 1989.

[31] H. C. Rao, Y. Chen, and M. Chen. A proxy-based personal web archiving service. *SIGOPS Oper. Syst. Rev.*, 35(1):61–72, 2001.

[32] V. Reich and D. S. Rosenthal. LOCKSS: A permanent web publishing and access system. *D-Lib Magazine*, 7(6), 2001.

[33] A. Ross. Internet Archive forums: Web forum posting. October 2004. http://www.archive.org/iathreads/post-view.php?id=23121.

[34] J. Rothenberg. *Avoiding Technological Quicksand: Finding a Viable Technical Foundation for Digital Preservation*. Number CLIR Report #77. Council on Library and Information Resources, Washington, DC, USA, 1999.

[35] Search engine partnership chart. http://www.ihelpyou.com/search-engine-chart.html.

[36] K. Shafer, S. Weibel, E. Jul, and J. Fausey. Persistent uniform resource locators. PURLs.

[37] J. A. Smith, F. McCown, and M. L. Nelson. Observed web robot behavior on decaying web subsites. *D-Lib Magazine*, 12(2), Feb 2006.

[38] D. Sullivan. Ending the debate over cloaking. February 2003. http://searchenginewatch.com/sereport/article.php/2165231.

[39] A. Swartz. arcget: Retrieve a site from the Internet Archive, Dec 2005. http://www.aaronsw.com/2002/arcget/.

[40] M. A. Veronin. Where are they now? A case study of health-related web site attrition. *Journal of Medical Internet Research*, 4(2), 2002.

[41] D. Waters and J. Garrett. Preserving digital information: Report of the task force on archiving of digital information. Technical report, 1996. http://www.rlg.org/ArchTF/.